VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza textu pomocí neuronových sítí

Text Analysis Using Neural Networks

2020 Bc. Marek Ulip

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student:	**Bc. Marek Ulip**
Studijní program:	N2647 Informační a komunikační technologie
Studijní obor:	2612T025 Informatika a výpočetní technika
Téma:	Analýza textu pomocí neuronových sítí Text Analysis Using Neural Networks
Jazyk vypracování:	čeština

Zásady pro vypracování:

Cílem práce je popsat metody pro analýzu topiků založené na hlubokém učení a neuronových sítí. Vybrané metody budou popsány, implementovány a ověřeny na vhodných datových sadách.

Práce bude obsahovat:
1. Popis metod analýzy topiků za využití neuronových sítí a jejich porovnání s klasickými metodami.
2. Popis vybraných algoritmů využívající neuronové sítě.
3. Návrh implementace a implementace algoritmů.
4. Experimentální ověření algoritmů a jejich porovnání s klasickými metodami.

Seznam doporučené odborné literatury:

[1] Chris Manning and Hinrich Schütze, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA: May 1999
[2] Dan Jurafsky: Speech and Language Processing, Pearson Education, 2014
[3] Bing Liu: Sentiment Analysis: Mining Opinions, Sentiments, and Emotions, Cambridge University Press; 1 edition (June 4, 2015)
[4] Charu C. Aggarwal, Machine learning for text. New York, NY: Springer Science+Business Media, 2018. ISBN 978-3-319-73530-6.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Jan Platoš, Ph.D.**

Datum zadání: 01.09.2019
Datum odevzdání: 30.04.2020

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 12. května 2020 .....

Rád bych na tomto místě poděkoval doc. Ing. Janu Platošovi, Ph.D. za odbornou pomoc a konzultace při vytváření této diplomové práce. Dále bych chtěl poděkovat mému otci za kontrolu pravopisu této práce a za jeho podporu.

Abstrakt

Tato diplomová práce se zabývá klasifikací textu a analýzou topiků v textech. Cílem bylo nejprve sepsat klasické a neuronové modely pro zmíněné disciplíny a tyto dvě skupiny metod experimentálně porovnat. Pro účely experimentů bylo získáno a popsáno celkem šest datových sad, z toho jedna byla v českém jazyce. Při experimentech s klasifikací bylo použito pět klasických modelů a sedm architektur neuronových sítí. V případě analýzy topiků byl porovnáván model LDA s experimentálním autoenkodérem, vytvořeným v rámci této práce, kde váhy naučeného autoenkodéru jsou použity jako model topiků. Výsledky ukazují, že v rámci klasifikace jsou klasické metody schopné dosahovat velmi dobré přesnosti, která se vyrovná neuronovým sítím, a to v mnohem kratším čase, avšak neuronové sítě jsou u většiny datových sad přesnější než klasické modely. V případě analýzy topiků vytvářel klasický model LDA kvalitnější topiky, avšak neuronový model byl v některých případech schopen lépe shlukovat analyzované dokumenty a kvalitou topiků nebyl na tolik vzdálen metodě LDA. Práce tedy představuje zajímavou alternativu ke klasickému modelu LDA.

Klíčová slova: strojové učení, hluboké učení, neuronové sítě, analýza textu, analýza topiků, topiky, autoenkodér, klasifikace textu

Abstract

This diploma thesis is concerning with text classification and text topic analysis. The aim was to first write and experimentally compare classic and neural models for mentioned disciplines. Total of six datasets were acquired for the purpose of experimentation. Five classical models and seven neural network architectures were used for the experiments. Model LDA and experimental autoencoder, created as part of this thesis which was using weights of learned autoencoder as topic model, were compared in case of topic analysis. The results show that in case of classification classical methods can achieve very good accuracy that can be compared to neural networks and with much less required time however neural networks are more accurate than classical models. In case of topic analysis, the classical model LDA was creating better topics however neural model was in some cases capable of better clustering of analyzed documents and its topic quality was not very distant from LDA method. This thesis introduces an interesting alternative to classical model LDA.

Keywords: machine learning, deep learning, neural networks, text analysis, topic analysis, topics, autoencoder, text classification

Obsah

Seznam použitých zkratek a symbolů **8**

Seznam obrázků **9**

Seznam tabulek **10**

1 Úvod **11**

2 Modely pro analýzu textu **12**
2.1 TF-IDF . . . 12
2.2 Latentní sémantická analýza . . . 12
2.3 Latent Dirchlet Allocation (LDA) . . . 15
2.4 Naivní Bayes (NB) . . . 16
2.5 Rozhodovací stromy . . . 16
2.6 Metoda podpůrných vektorů . . . 18

3 Neuronové sítě **19**
3.1 Aktivační funkce . . . 19
3.2 Ztrátová funkce . . . 20
3.3 Optimalizéry . . . 20
3.4 Regularizace . . . 21
3.5 Hustá síť . . . 21
3.6 Rekurentní síť . . . 22
3.7 Konvoluční síť . . . 24
3.8 Vnoření slov . . . 27
3.9 Autoenkodér . . . 28
3.10 TensorFlow a Keras . . . 29

4 Měřítka kvality modelů **30**
4.1 Klasifikace . . . 30
4.2 Kvalita modelu topiků . . . 30

5 Datové sady **31**
5.1 Reuters . . . 31
5.2 20 Newsgroups . . . 31
5.3 DBpedia Ontology Classification Dataset . . . 31
5.4 AG's News Topic Classification Dataset . . . 32
5.5 Yelp . . . 32
5.6 Česko-Slovenská filmová databáze - ČSFD . . . 32

6 Klasické modely - experimenty **34**
6.1 Postup testování 34
6.2 Vliv způsobu předzpracování na přesnost modelu 36
6.3 Výsledky testů 37

7 Neuronové sítě - experimenty **46**
7.1 Výsledky testů 46
7.2 Porovnání s klasickými metodami 49
7.3 Využití vnořené vrstvy 50

8 Extrakce topiků pomocí autoenkodéru **51**
8.1 Základní model 51
8.2 Získání topiků 51
8.3 Porovnání s metodou LDA 56
8.4 Výsledky u ostatních datových sad 56
8.5 Výsledky na české datové sadě 58
8.6 Shlukování dokumentů 58

9 Závěr **60**

10 Přílohy **64**

Seznam použitých zkratek a symbolů

LDA	–	Latent Dirchlet Allocation
LSA	–	Latent Semantic Analysis
SVD	–	Singular Value Decomposition
API	–	Application Programming Interface
TF	–	Term Frequency
SVM	–	Support Vector Machine
RNN	–	Recurrent Neural Network
LSTM	–	Long Short Term Memory
NB	–	Naive Bayes
GRU	–	Gated Recurrent Unit
DT	–	Decision Tree
RF	–	Random Forest

Seznam obrázků

1 Ukázka dělící přímky SVM 18
2 Průchod jednoduchou RNN (Christopher Olah 2015) 22
3 LSTM Buňka (Christopher Olah 2015) 24
4 Konvoluční síť (Zhang, Y., & Wallace, B. (2015)) 26
5 Architektura autoenkodéru (Scientific Figure on ResearchGate 2018) 28
6 Vliv předzpracování podle modelu u datové sady AGs News 37
7 Vliv odstranění krátkých článků na přesnost 38
8 Naměřené přesnosti jednotlivých modelů pro datovou sadu Reuters 39
9 Naměřené přesnosti jednotlivých modelů pro datovou sadu AG's News 39
10 Naměřené přesnosti jednotlivých modelů pro datovou sadu DBpedia 40
11 Naměřené přesnosti jednotlivých modelů pro datovou sadu 20 Newsgroups 41
12 Matice záměn pro model Naive Bayes 43
13 Naměřené přesnosti jednotlivých modelů pro datovou sadu Yelp 44
14 Matice záměn pro nevyváženou datovou sadu ČSFD (pro model SVM) 45
15 Architektura autoenkodéru pro 4 témata 52
16 Ukázka kvality slov při vysoké kvalitě shlukování a nízkém skóre koherence a opačně vysokém skóre koherence a nízké kvalitě shlukování. 53
17 Ukázka topiků neuronového modelu a modelu LDA pro sadu AGs News 57
18 Ukázka topiků neuronového modelu a modelu LDA pro sadu ČSFD 58

Seznam tabulek

1 Nejlepší a nejhorší přesnost Naive Bayes při různých metodách předzpracování . 36
2 Přesnost jednotlivých modelů pro datovou sadu Reuters 38
3 Přesnost jednotlivých modelů pro datovou sadu AGNews 39
4 Přesnost jednotlivých modelů pro datovou sadu DBpedia 40
5 Přesnost jednotlivých modelů pro datovou sadu 20 Newsgroups 41
6 Přesnost jednotlivých modelů pro datovou sadu 20 Newsgroups se 7 tématy . . . 42
7 Přesnost jednotlivých modelů pro datovou sadu Yelp 42
8 Přesnost jednotlivých modelů pro datovou sadu ČSFD (bez předzpracování) . . . 44
9 Přesnost jednotlivých modelů pro datovou sadu ČSFD (s předzpracováním) . . . 45
10 Přesnost jednotlivých neuronových sítí pro datovou sadu Reuters 46
11 Přesnost jednotlivých neuronových sítí pro datovou sadu AG's News 47
12 Přesnost jednotlivých neuronových sítí pro datovou sadu DBpedia 47
13 Přesnost jednotlivých neuronových sítí pro datovou sadu 20 Newsgroups 48
14 Přesnost jednotlivých neuronových sítí pro datovou sadu Yelp 48
15 Přesnost jednotlivých neuronových sítí pro datovou sadu ČSFD 49
16 Porovnání přesností klasických modelů s modely neuronových sítí vzhledem k datové sadě 49
17 Vliv počtu epoch na kvalitu modelu topiků 54
18 Vliv regularizace aktivační funkce na kvalitu topiků 55
19 Vliv regularizace kernelu na kvalitu topiků 55
20 Vliv reprezentace dokumentů pomocí TFIDF 56
21 Porovnání LDA s neuronovým modelem topiků 56
22 Kvalita modelu u ostatních datových sad 57

1 Úvod

Obsahem této práce je popis nejčastěji používaných metod pro analýzu topiků v textech a klasifikace dokumentů. Dále je u popsaných metod testována jejich přesnost a vliv předzpracování na přesnost. Pro experimenty byly použity již existující knihovny, které dané metody obsahují. Součástí práce bylo i nalezení a popsání datových sad, na kterých byly zmíněné testy provedeny.

Další oblastí, kterou se tato práce zabírá, je extrakce topiků pomocí neuronových sítí, k čemuž se v současnosti nejčastěji používá metoda LDA.

Práce je dělena na teoretickou a praktickou část. Kapitoly 2 až 4 patří do teoretické části a slouží pro snazší pochopení metod použitých v následující praktické části.

V kapitole 2 jsou uvedeny klasické metody používané pro klasifikaci. Je zde také popsán model pro extrakci topiků LDA, který byl testován i na přesnost v klasifikaci. V další kapitole jsou popsány neuronové sítě použitelné pro klasifikaci textu. Dále je popsán autoenkodér, který byl v rámci této práce použit pro extrakci topiků. Nakonec je zmíněna knihovna TensorFlow a API vytvořené pro snazší vytváření neuronových síti Keras. Kapitola 4 popisuje metriky, které byly použity při analýze jednotlivých modelů.

První kapitola praktické části s číslem 5 popisuje analyzované datové sady, jaké byly použity metody předzpracování, a jak obsáhlé jsou tyto datové sady. Kapitola 6 popisuje postup testování a výsledky testů klasických metod popsaných v kapitole 2. Obdobně kapitola 7 popisuje postup a výsledky testů na modelech neuronových sítí. Nakonec kapitola 8 popisuje extrakci topiků pomocí neuronové sítě. Ukazuje, jaké metody zlepšuji kvalitu topiků, kvalitu výsledných topiků a také srovnání s modelem LDA.

2 Modely pro analýzu textu

Testy byly provedeny na modelech Naive Bayes, Latent Semantic Analysis (LSA), Latent Dirchlet Allocation (LDA), metody podpůrných vektorů (SVM), rozhodovací strom a náhodný les. V této kapitole je také popsána metodika TF-IDF, která se používá v kombinaci se zmíněnými modely za účelem normalizace a zvýšení přesnosti.

2.1 TF-IDF

TF-IDF [30] je metoda pro určení významnosti jednotlivých slov v dokumentech. Využívá při tom počtu slov, označováno jako četnost slova (anglicky Term Frequency - TF), a podle četnosti výskytu jednotlivých slov napříč datovou sadou se určuje významnost daných slov, což je označováno jako převrácená četnost slova ve všech dokumentech (anglicky Inverse Document Frequency - IDF).

TF zjišťuje počet výskytu daného slova v dokumentu. Může to být přímo počet, nebo se využívá normalizace pro snížení významu u slov, které se vyskytují až příliš často.

IDF určuje významnost daného slova v dokumentech. Slova, která se vyskytují jen v minimu dokumentech jsou pravděpodobně svázána s určitým tématem a mohou mít vysokou informační hodnotu, proto jim bude přiděleno vyšší číslo. Na druhou stranu slovo, které se vyskytuje v mnoha dokumentech (typicky se jedná o stopslova, spojky, předložky, zájmena atp.) bude mít minimální význam, a proto mu bude přiděleno malé číslo. Matematicky se tato významnost vypočte jako $log(N/df(t))$, kde `N` je celkový počet dokumentů v korpusu a `df(t)` určuje počet dokumentů, ve kterých se vyskytuje slovo `t`.

Výsledkem obou metod jsou tabulky, kde sloupce jsou jednotlivé dokumenty a řádky jednotlivá slova. TF-IDF spojuje tyto dvě metody vynásobením buněk zmíněných tabulek mezi sebou - TF*IDF. Nově získaná tabulka bude mít větší hodnoty u slov, které se vyskytují v méně dokumentech a zároveň unikátním slovům (například překlepy) přidělí nízkou hodnotu díky tomu, že mají malou četnost.

2.2 Latentní sémantická analýza

Metoda TF-IDF určila významnost jednotlivých slov, ale má několik problémů. Výsledná matice má velké rozměry, a navíc může obsahovat slova, která mají podobný význam, nebo mají více než jeden význam. Metoda latentní sémantické analýzy (anglicky Latent Semantic Analysis - LSA) tento problém řeší redukcí dimenze pomocí metody singulárního rozkladu (SVD) a veškerou analýzu poté provádí na této redukované matici.

2.2.1 Singulární rozklad

Singulární rozklad (anglicky Singular Value Decomposition - SVD) pomáhá nalézt koncepty (což jsou v případě této práce témata) v matici TF-IDF. Po aplikaci SVD je původní matice o rozměrech $m \times n$ vyjádřena jako součin tří matic. Tento vztah je zapsán jako $M = U\Sigma V^*$, kde M je původní matice (např. TF-IDF). U je matice o rozměrech $m \times m$, a dá se na ní dívat jako na matici, která vyjadřuje vztah mezi slovem a konceptem (jak moc je dané slovo významné pro daný koncept (téma)). Σ je diagonální matice o rozměrech $m \times n$, a vyjadřuje významnost jednotlivých vektorů (sloupců u matice U a řádků u matice V^*), kde významnost je setříděna od nejvyššího po nejnižší. V je matice o rozměrech $n \times n$, která vyjadřuje vztah mezi článkem a konceptem (jak moc daný článek k jednotlivým konceptům náleží). Tyto matice je poté možné redukovat tak, že se vezme k sloupců z matice U a k řádků z matice V^*. Hodnota k se vypočítává pomocí matice Σ, ze které se ubírají hodnoty (od konce) dokud je celkový čtvercový součet zbylých hodnot větší než 80-90 % (doporučené rozmezí) původní Σ . Pokud by se tyto redukované matice znovu vynásobily, výsledná matice by se podobala původní matici M.

Pro představu mějme matici M uvedenou níže, kde jednotlivé řádky představují slova, sloupce jsou dokumenty a hodnoty buňky vyjadřují, kolikrát se dané slovo vyskytlo v daném dokumentu - nejjednodušší verze TF matice. Tato matice obsahuje 5 dokumentů, kde napříč všemi dokumenty bylo použito 8 unikátních slov. [1]

$$\begin{bmatrix} 2 & 4 & 5 & 0 & 0 \\ 5 & 3 & 2 & 0 & 0 \\ 4 & 1 & 7 & 0 & 0 \\ 5 & 2 & 5 & 0 & 0 \\ 0 & 2 & 0 & 3 & 4 \\ 0 & 0 & 0 & 6 & 4 \\ 0 & 3 & 0 & 2 & 5 \\ 0 & 1 & 0 & 8 & 8 \end{bmatrix}$$

Matice je po aplikaci metody SVD rozložena na součin tří matic, uvedených níže, kde, jak již bylo zmíněno, sloupce první matice U vyjadřují, jak je dané slovo významné pro dané téma, nenulové hodnoty druhé matice Σ určuje sílu jednotlivých témat - jak významné jsou jednotlivé sloupce první matice a jednotlivé řádky třetí matice V^* pro dané téma. Vztah mezi jednotlivými buňkami z matice Σ a řádky a sloupci ze zbylých dvou je barevně zvýrazněn. Nakonec řádky třetí matice vyjadřují, jak moc dokument náleží do jednotlivých témat.

[1] Hodnoty byly vymyšleny pro jednodušší demonstraci, věty by ve skutečnosti nedávaly moc smyslu.

$$U = \begin{bmatrix} 0.1199 & 0.4445 & 0.2761 & -0.5228 & 0.6188 \\ 0.0999 & 0.3902 & 0.3275 & 0.7207 & 0.1623 \\ 0.1068 & 0.5658 & -0.4519 & -0.2860 & -0.3785 \\ 0.1115 & 0.5259 & -0.1091 & 0.2495 & -0.1743 \\ 0.3395 & -0.0435 & 0.2669 & -0.0775 & -0.0037 \\ 0.4378 & -0.1307 & -0.3778 & 0.1312 & 0.4619 \\ 0.3569 & -0.0192 & 0.5897 & -0.1946 & -0.4391 \\ 0.7192 & -0.1842 & -0.1961 & 0.0442 & -0.1039 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 15.3588 & 0 & 0 & 0 & 0 \\ 0 & 13.4475 & 0 & 0 & 0 \\ 0 & 0 & 4.4233 & 0 & 0 \\ 0 & 0 & 0 & 3.3764 & 0 \\ 0 & 0 & 0 & 0 & 1.8177 \end{bmatrix}$$

$$V^* = \begin{bmatrix} 0.1122 & 0.2330 & 0.1370 & 0.6584 & 0.6933 \\ 0.5750 & 0.3151 & 0.7134 & -0.1805 & -0.1686 \\ -0.0369 & 0.7966 & -0.3782 & -0.4196 & 0.2115 \\ 0.7881 & -0.1217 & -0.5709 & 0.1537 & -0.1198 \\ -0.1847 & 0.4437 & -0.0560 & 0.5779 & -0.6570 \end{bmatrix}$$

Matice U a V^* lze redukovat pomocí matice Σ, kde každá hodnota představuje váhu možného tématu. Předpokládejme, že víme, že dokumenty spadají do 3 různých témat, avšak matice Σ má 5 hodnot. Můžeme tedy poslední 2 hodnoty vynechat. Na matici U se to projeví tak, že budeme pracovat pouze s prvními 3 sloupci a analogicky v matici V^* budeme pracovat pouze s prvními 3 řádky.

Po provedené redukci lze použít redukované matice pro práci s neviděnými dokumenty. Chceme zjistit téma daného dokumentu, pro což se nejlépe hodí matice U. Neviděný dokument je reprezentován formou vektoru, kde každá z jeho položek odpovídá počtu daného slova v dokumentu. Pořadí a počet slov se musí shodovat s pořadím a počtem slov u TF-IDF matice.

Vektor neviděného dokumentu je zobrazen níže. [2]

$$\begin{bmatrix} 2 & 1 & 0 & 1 & 5 & 0 & 2 & 8 \end{bmatrix}$$

Tento vektor vynásobíme s maticí U, po čemž vznikne vektor uvedený níže, kde jednotlivá čísla vyjadřují, jak moc daný dokument náleží do daného tématu. Je vidět, že téměř určitě bude zkoumaný dokument hovořit o prvním tématu.

$$\begin{bmatrix} 8.6161 & 0.0756 & 1.7157 \end{bmatrix}$$

[2] Vektor opět obsahuje smyšlené hodnoty pro snazší demonstraci.

Avšak zde se projevuje jedna z nevýhod SVD, a to, že lze jen těžko určit, jaká témata jednotlivé koncepty reprezentují. Na druhou stranu je tato metoda velmi efektivní a dokáže pracovat s velmi velkými maticemi a pro většinu datových sad podává dobré výsledky.

Mnoho matematických faktů ohledně SVD bylo vypuštěno pro snazší pochopení aplikace SVD při analýze textových dat. Metoda je detailněji popsaná v knize Machine Learning for Text [2].

2.3 Latent Dirchlet Allocation (LDA)

LDA [31] využívá při práci s dokumenty pravděpodobnost. Vychází z předpokladu, že každý dokument byl vytvořen výběrem z daného počtu témat s určitou pravděpodobností pro každé téma. Každé téma pak bylo vytvořeno výběrem daného počtu slov z daného tématu s určitou pravděpodobností pro každé slovo.

2.3.1 Vznik dokumentu

Předpokládejme, že dokument bude vytvořen ze 3 témat (technika, ekonomika, politika), kde pravděpodobnosti pro jednotlivé témata jsou (70 %, 20 %, 10 %). Dále předpokládejme, že každé téma má 3 slova s pravděpodobnostmi (40 %, 30 %, 30 %). Pro vytvoření dokumentu o určitém počtu n slov se bude n-krát na základě pravděpodobnosti jednotlivých témat vybírat téma a po výběru tématu se na základě pravděpodobnosti jednotlivých slov vybere jedno slovo z daného tématu. Například první slovo může ze 70 % patřit do tématu technika a ze 40 % to může být první slovo z tohoto tématu.

2.3.2 Popis algoritmu

Aby mohl být algoritmus spuštěn, musí dopředu znát počet témat, která se v dokumentech mohou vyskytnout.

1. Nejdříve se náhodně přidělí každému slovu v každém dokumentu jedno z témat.
2. Pro každý dokument `d`
   (a) Vypočítá dva poměry (zároveň předpokládá, že přidělená témata u ostatních dokumentů jsou správně)
      i. Poměr slov v dokumentu `d`, která jsou současně přidělena tématu t, kde $t = p(temat/dokumentd)$
      ii. Poměr přidělení tématu `t` danému slovu, napříč všemi dokumenty, kde poměr ve se vypočte jako $w = p(slovow/temat)$
   (b) Dva získané poměry `t` a `w` se mezi sebou vynásobí a výsledek je přidělen právě zpracovávanému slovu. Téma slova $= t \times w$

3. Bod 2 se buď po stanovenou dobu opakuje, nebo pracuje, dokud nejsou přiřazení konzistentní.

2.4 Naivní Bayes (NB)

Naivní Bayes také využívá k přiřazování tématu pravděpodobnost, a to pravděpodobnost každého atributu (v našem případě slova), s jakou patří do jednotlivých tříd (témat). Je zřejmé, že pro každou třídu se bude pravděpodobnost atributu lišit s počtem výskytu v dané třídě. Algoritmus vychází z Bayesova teorému a říká se mu naivní, protože 'naivně' předpokládá, že pravděpodobnost, s jakou daný atribut patří do dané třídy, je nezávislá na všech ostatních atributech. Vliv tohoto předpokladu na výslednou přesnost není až tak velký, a navíc je díky tomu algoritmus velmi rychlý a efektivní. V případě práce s textovými dokumenty se spolu s pravděpodobností jednotlivých slov bere i ohled na jejich množství - Naive Bayes využívá multinomiální model.

Naive Bayes využívá Bayesův teorém $P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$. Při práci s klasifikací textu by použití tohoto vzorce mohlo vypadat například jako $P(Automobily|Auto\,jelo\,velmi\,pomalu)$, což se dá přečíst jako pravděpodobnost, že článek mluví o automobilech, jestliže používá daná slova (Auto jelo velmi pomalu). Tuto pravděpodobnost musíme vypočíst pro všechna témata, se kterými pracujeme. Pro každé téma chceme vypočítat $\frac{P(Auto\,jelo\,velmi\,pomalu|Tema) \times P(Tema)}{P(Auto\,jelo\,velmi\,pomalu)}$.

Pro výpočet $P(Auto\,jelo\,velmi\,pomalu|Tema)$ je třeba znát počet výskytu celého výrazu 'Auto jelo velmi pomalu', ale celý tento výraz se v datové sadě nejspíše vyskytovat nikdy nebude, proto se musí vypočítat pravděpodobnost, se kterou jednotlivá slova patří do daného tématu - $P(Auto|Tema)$, $P(jelo|Tema)$ a tak dále.

Ve výsledku se vypočítá $P(Auto\,jelo\,velmi\,pomalu|Tema) = P(Auto|Tema) \times P(jelo|Tema) \times P(velmi|Tema) \times P(pomalu|Tema)$. Právě díky naivnímu předpokladu, že jsou pravděpodobnosti výskytu jednotlivých slov na sobě nezávislé, můžeme tyto pravděpodobnosti násobit. Často ale nastane situace, kdy dané slovo v tématu není, pravděpodobnost by tedy byla nulová a tím by byla ztracena kompletní informace o pravděpodobnosti, že článek patří do určitého tématu (většinou by pro každý článek vyšla 0). Proto se využívá tzv. "Laplace Smoothing", který zajistí, že pravděpodobnost bude vždy nenulová. Po získání pravděpodobností pro všechna témata se tyto pravděpodobnosti porovnají a téma s největší pravděpodobností bude odhadovaným tématem pro daný článek.

Opět se jedná pouze o stručný popis tohoto modelu. Více informací o této metodě lze nalézt v knize Machine Learning for Text [2].

2.5 Rozhodovací stromy

Rozhodovací stromy [20] jsou často používány pro vytvoření klasifikačních modelů, jelikož pravidla, která si tento model vytváří, jsou si podobná s tím, jak člověk přemýšlí a jsou snadno interpretovatelná. Jsou to sekvenční modely, které logicky kombinují sekvenci jednoduchých

testů, kde každý z testů pracuje s jedním atributem datové sady a buď porovnává, zda-li numerická hodnota přesahuje určitý práh, nebo, v případě kategorických dat, zda-li hodnota spadá do určité množiny povolených hodnot. Tato pravidla lze velmi snadno interpretovat na rozdíl od vah v neuronových sítích.

Při učení se model snaží v datech nalézt vzory a to tak, že rozhoduje, které testy na jednotlivých atributech nejlépe rozdělují dané položky do rozdílných tříd. Rozhodnutí o nejlepším možném rozdělení se provádí na základě nejlepší hodnoty dané rozdělovací metriky, jako například gini index nebo entropie. Rozhodovací pravidla mohou využít stejný atribut opakovaně (například data, která měla velikost atributu menší než 0,5, mohou být poté dále rozdělena např. na základě hodnoty 0,2). Rozdělování se provádí do té doby, dokud každý uzel nezastupuje pouze jednu danou třídu (prvky už v uzlu nejsou promíchané). Jakmile je strom sestaven, měl by mít 100 % přesnost na trénovací části dat, ale pokud je daný strom příliš složitý, většinou to vede k horšímu zobecňování, a tudíž i horší přesnosti u testovací části dat.

Pro zlepšení zobecňovací schopnosti stromu se používá tzv. prořezávání, které se snaží vytvořit podstrom z původního stromu, který zamezuje přeučení u trénovací části dat. Tomuto prořezávání se říká také následné prořezávání a to proto, že i při vytváření stromu dochází k určitému před prořezání, kdy například nový uzel nemůže vzniknout, pokud v sobě nemá určitý počet prvků, nebo rozhodnutí řezu je provedeno na příliš malém vzorku. Existuje více technik prořezávání, kde většina z nich prochází uzly buď odspodu nahoru, nebo odshora dolů. Uzel je odstraněn, pokud jeho odstranění zlepší dané kritérium.

2.5.1 Náhodný les

Náhodný les [32] (v angličtině Random Forest nebo také random decision forest) je sestaven z množství jednotlivých rozhodovacích stromů, které dohromady tvoří jeden celek. Každý strom samostatně dává svůj výsledek a výsledek, který se vyskytuje nejčastěji, je poté označen za výsledek náhodného lesa. Tento model je založen na konceptu víc hlav víc ví, kde velké množství relativně nesouvisejících modelů, které pracují jako celek, je schopno překonat kteréhokoliv z individuálních členů celku.

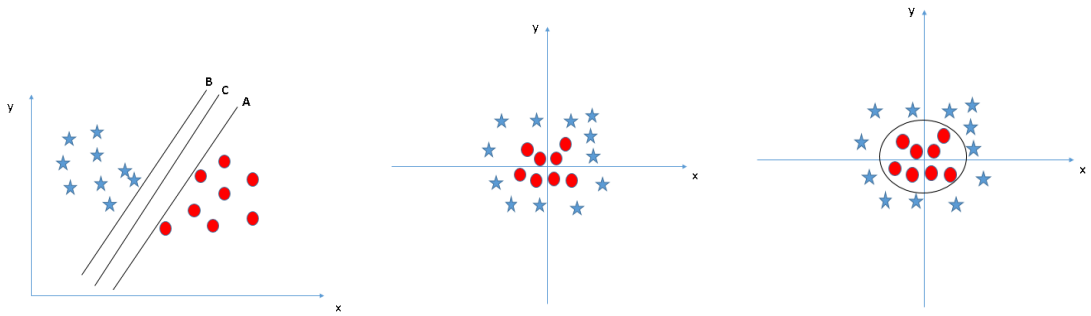
Při trénování tohoto modelu se na začátku náhodně vybere určitý počet prvků, který vytvoří trénovací sadu pro jeden strom v lesu. Poté se na základě předem určeného počtu m náhodných atributů, který je menší, než celkový počet atributů, vybere takový výběr, jehož rozdělení nejvíce přispěje k rozdělení trénovacích dat. Hodnota m se během trénování lesa nemění. Žádný ze stromů v lese není prořezáván a je trénován do největšího možného rozměru.

Bylo ukázáno [4], že poměr chyb náhodného lesa spočívá v korelaci mezi jednotlivými stromy, kdy čím větší korelace, tím horší je výsledná přesnost lesa. Dalším faktorem, ovlivňujícím přesnost, je přesnost jednotlivých stromů, kdy zvyšování přesnosti jednotlivých stromů zvyšuje celkovou přesnost náhodného lesa. Tyto dva zmíněné faktory jsou silně ovlivněny volbou hodnoty m, kdy snižování této hodnoty snižuje jak korelaci, tak přesnost, a naopak zvýšení této hodnoty oba zmíněné faktory zvyšuje.

2.6 Metoda podpůrných vektorů

Metoda podpůrných vektorů (anglicky Support Vector Machines) [22] je model, který pří klasifikaci využívá nadprostoru (prostor, jehož dimenze je o jednu menší, než dimenze dat se kterými se pracuje. Například u dvourozměrného prostoru se jedná o přímku a u trojrozměrného o plochu), který rozděluje data na dvě třídy. Učení tohoto modelu spočívá v nalezení optimálního nadprostoru. Pro jednoduchost bude dále uváděn příklad nad dvourozměrnými daty. Pro nalezení optimální přímky oddělující dvě třídy se nejdříve vytvoří u každé z tříd vektory, které jsou nejblíže potencionální oddělující čáře (tyto vektory jsou na okraji jednotlivých tříd). Těmto vektorům se říká podpůrné a na základě jejich vzdálenosti se volí optimální oddělující přímka, kdy se hledá co největší vzdálenost mezi oběma třídami a co největší vzdálenost každé třídy od dělící přímky. Tento princip je uveden na obrázku 1a. V jednoduchosti se SVM snaží najít přímku, která leží v ideálním středu dvou tříd.

V případě lineárně neseparabilních problémů už přímka jako oddělující rovina nestačí. V tomto případě převede model data do vyššího rozměru, ve kterém již jsou lineárně separabilní. Na obrázku 1b je příklad lineárně neseparabilní množiny. SVM přidá osu z a data na této ose zobrazí jako $z = x^2 + y^2$. Při zobrazení dat pomocí této osy jsou již data lineárně separabilní. Po nalezení vhodné přímky se určí že oddělující přímka je $z = k$ a po dosazení za z v původní rovnici dostaneme $k = x^2 + y^2$, což je rovnice kruhu, který po převedení dat zpátky na dvourozměrná dokáže data oddělit tak, jak je vidět na obrázku 1c. Ne vždy jsou tyto transformace tak snadné a existuje mnoho přístupů k jejich nalezením. Obecně se označují jako kernel trick.



(a) Princip nalezení přímky SVM (Sunil Ray 2017)

(b) Ukázka lineárně neseparabilní množiny (Sunil Ray 2017)

(c) Ukázka oddělení lineárně neseparabilní množiny (Sunil)

Obrázek 1: Ukázka dělící přímky SVM

3 Neuronové sítě

Neuronové sítě se řadí do podkategorie strojového učení tzv. 'hluboké učení' (anglicky Deep Learning). Zjednodušeně je to model obsahující mnoho vrstev, kde každá vrstva slouží jako filtr, kdy po průchodu všemi filtry jsou data 'očištěná' a lépe vhodná pro další analýzu. Existuje velmi mnoho vrstev a v rámci analýzy textu budou zmíněny husté, rekurentní a konvoluční vrstvy. Neuronové sítě často při práci s textem používají tzv. 'vnoření slov' (anglicky Word Embedding), které zde budou také zmíněné.

3.1 Aktivační funkce

Aktivační funkce [1] přímo ovlivňují, jaký bude výstup neuronové sítě. Je aplikována na každý neuron v síti a na základě jejího výsledku se rozhoduje, zda-li se má neuron aktivovat - udává, je-li daný neuron důležitý pro predikci aktuálního vstupu. Aktivační funkce také pomáhá normalizovat výstup každého neuronu do rozmezí 0 až 1 nebo -1 až 1. Aktivační funkce musí být výpočetně efektivní, protože se počítají přes tisíce až miliony neuronů pro každý vstupní prvek. Níže jsou z mnoha existujících aktivačních funkcí zmíněny ty, které byly použity při analýze v sekci 7.

3.1.1 ReLu

Tato funkce je nelineární variantou lineární aktivační funkce, která není vhodná při učení neuronových sítí, protože u ní nelze spočítat derivaci (je nespojitá v nule), která je při hledání optimálních vah nezbytná. ReLu je modifikována tak, že je již spojitá, a tudíž je možné u ni spočítat derivaci. Tato funkce vrací pro záporná čísla 0 a pro kladná je shodná s lineární funkcí. Výhodou této funkce je, že je výpočetně efektivní a umožňuje síti rychle konvergovat. Není ovšem vhodná pro vstupy, blížící se nule, nebo záporné vstupy, u kterých je nulová derivace a s touto hodnotou není možné provádět učení.

3.1.2 Sigmoid

Aktivační funkce, která má výstupy v rozsahu od nuly do jedné a její tvar se podobá písmenu S. Výhodou je, že má rozsah 0-1, a tím je zajištěno že je její výstup normalizován. Dále má hladký gradient, čímž se zabraňuje výkyvům ve výstupních hodnotách. Nakonec má jasné predikce, kdy hodnoty nad 2 nebo pod -2 budou jako výstup mít hodnoty blízké 1 nebo 0. Nevýhodami jsou, že tato funkce trpí problémem mizejícího gradientu pro příliš vysoké nebo nízké hodnoty, což může způsobit, že se síť přestane učit nebo se k výsledku dobere velmi pomalu. Dále je tato funkce výpočetně náročná a není centrována kolem nuly.

3.1.3 Softmax

Softmax je zobecněná sigmoida, která umožňuje pracovat s více třídami a to tak, že normalizuje výstupy do rozmezí 0 až 1 a tuto hodnotu vydělí jejich součtem, čímž udává, s jakou pravděpodobností hodnota náleží určité třídě. Tato funkce se typicky používá pouze pro výstupní vrstvu, kde je třeba klasifikovat vstupy do více kategorií.

3.1.4 Hyperbolický tangent

Místo hyperbolického tangentu (TanH) se v dnešní době používá ReLu funkce. Avšak TanH je v knihovně Keras nastaven jako výchozí aktivační funkce pro rekurentní sítě, a proto je zde zmíněna. Je centrována kolem 0, díky čemuž je možné modelovat vstupy, které mají vysoce záporné, nulové nebo vysoce kladné hodnoty. Jinak se výhody a nevýhody této funkce podobají výhodám a nevýhodám funkce sigmoid 3.1.2 .

3.2 Ztrátová funkce

Ztrátová funkce [14] je metoda pro určení, jak dobře daný model pracuje s daným problémem. Čím lépe model chápe daný problém, tím menší bude chyba ztrátové funkce a naopak. V základu se ztrátové funkce dělí na regresní a klasifikační. Tato práce s regresí nepracuje, a proto budou zmíněny jenom ty funkce, které pracují s klasifikací.

3.2.1 Binární křížová entropie

Tato funkce se používá u binárních problémů například u analýzy sentimentu, kde je cílem určit, je-li text negativní nebo pozitivní. Tato funkce se často používá spolu se sigmoidou, která je v poslední vrstvě sítě. Tato funkce se nemusí používat jenom pro klasifikaci. Další využití má také u autoenkodérů obrázků.

3.2.2 Kategorická křížová entropie

Na rozdíl od binární křížové entropie se tato funkce používá pro řešení více třídových problémů. Nejčastěji se tato funkce používá spolu s funkcí softmax v poslední vrstvě klasifikační sítě.

3.3 Optimalizéry

Optimalizéry [26] jsou v neuronových sítích metody, které mění váhy sítě tak, aby výsledná chyba ve ztrátové funkci byla co nejmenší. Každý optimalizér mění váhy jinak, některé ještě navíc mění míru učení anebo zahrnují setrvačnost. Míra učení udává, jak rychle bude optimalizér konvergovat k minimu. Velké hodnoty způsobí rychlejší učení, ale optimalizér může přeskočit lokální nebo globální minimum. Na druhou stranu při nízkých hodnotách se učí pomalu a může se stát, že optimalizér uvízne v lokálním minimu. Setrvačnost je technika, která částečně řeší problém uvíznutí v lokálním minimu, kdy dostane-li se optimalizér do lokálního minima, tak

technika setrvačnosti mu umožní dál optimalizovat, i když bude chyba větší v naději, že v okolí minima existuje minimum větší než právě nalezené. V současnosti je nejpoužívanějším optimalizérem Adam.

3.4 Regularizace

Regularizace [28] jsou techniky, které provádí malé modifikace učícího algoritmu tak, aby model lépe zobecňoval a tím i měl lepší výsledky na doposud neviděných datech. V neuronových sítí regularizace penalizuje hodnoty ve váhové matici.

3.4.1 Regularizace L1 a L2

Tyto techniky regularizace penalizují příliš vysoké váhy. Upravují ztrátovou funkci tak, že přidají regularizační výraz. Přidáním tohoto výrazu se hodnoty váhové matice zmenší, protože se předpokládá, že síť s menšími váhovými maticemi vede k jednodušším modelům a tím se i sníží dopad přeučení.

3.4.2 Dropout

Dropout technika při trénování neuronové sítě odstraní určité předem dané procento neuronů spolu s jejich propojeními, takže každá iterace má jinou sadu uzlů a jiné výstupy. Dropout se hodí spíše u větších sítí, kde může více docházet k přeučení a dropout zavádí více náhody.

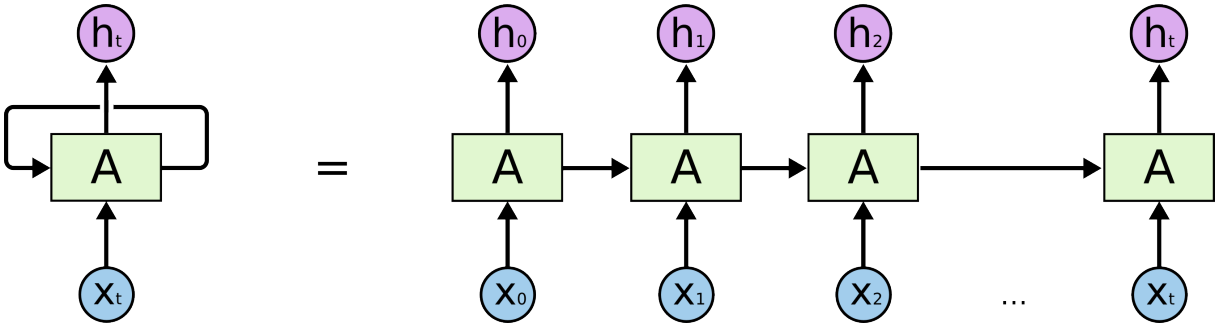
3.4.3 Early Stopping

Early stopping je technika, kdy se učení po určitém kritériu zastaví. Kritérium může být zhoršující se přesnost po jednotlivých epochách, nesnižující se ztrátová funkce a podobně. Zde je důležité dobře odhadnout, kolik epoch už má být považováno za nežádoucí a zastavit učení. Po zastavení učení je možné se rozhodnout, zda-li vrátit váhy do stavu, kdy síť měla nejlepší výsledky, anebo je ponechat tak, jak jsou.

3.5 Hustá síť

Hustá sít (anglicky Dense network) [21] je síť složená z mnoha jednoduchých perceptronů, které jsou všechny navzájem propojené mezi jednotlivými vrstvami. Perceptron je lineární model, který aplikuje na vstup naučené váhy, tento výsledek vloží do aktivační funkce a výsledek aktivační funkce je poté výstupem daného perceptronu. Často se ke vstupu také přidává bias, ale existují i případy, kdy se nepoužívá anebo je součásti vah.

Výhodou této sítě je, že se učí charakteristiky z kombinace všech charakteristik z předchozí vrstvy. Na druhou stranu kvůli velkému množství propojení je velmi výpočetně náročné danou síť naučit. Hustá síť se často používá jako výstupní vrstva u klasifikačních problémů.



Obrázek 2: Průchod jednoduchou RNN (Christopher Olah 2015)

3.6 Rekurentní síť

Jako většina neuronových sítí i rekurentní síť našla uplatnění v mnoha odvětvích jako klasifikace, zpracování mluveného slova, modelování jazyka, překlady, popisování obrázků, predikce hodnot (počasí, burza) a další. Pro tuto síť je charakteristické, že je velmi citlivá na pořadí, v jakém se vstupy do sítě dostanou. Je to díky tomu, že obsahuje paměť, kdy si pamatuje, jaké vstupy byly před daným vstupem a tuto informaci využívá pro určení výstupu. Dalším rozdílem oproti ostatním sítím je, že umí pracovat s proměnlivou velikostí vstupů a výstupů.

3.6.1 Jednoduchá RNN

Jednoduchá RNN [15] (v angličtině Simple RNN nebo též Vanilla RNN) tvoří základ, na kterém byly postaveny všechny pokročilejší RNN sítě. V základu pracuje podobně jako všechny ostatní neuronové sítě - dostanou vstupní vektor a vyprodukují výstupní vektor. Ale obsah výstupního vektoru je ovlivněn nejen vstupem a aplikovanými váhami, ale také celou historií předchozích vstupních vektorů. Tato historie je reprezentována jako vnitřní stav RNN sítě, který je upraven po každém vstupu. Tento vnitřní stav je v nejjednodušší podobě reprezentován jako vektor. Tento vektor před prvním vstupem obsahuje samé nuly. Po prvním vstupu je tento vektor přepočítán pomocí aktivační funkce, což u jednoduché RNN většinou bývá hyperbolický tangens (tanh). Tato aktivační funkce se stará o to, aby hodnoty ve stavovém vektoru byly v rozmezí -1 až 1. Přesněji nový stav je vypočítán jako: $h_t = tanh(W_h h h_t - 1 + W_x h x_t)$, kde $h_t - 1$ je stav vypočítán při předchozím vstupu, x_t je současný vstup a $W_h h$, $W_x h$ jsou váhové matice. Výstup je poté vypočten jako $W_h y h_t$. A tento postup je opakován pro všechny vstupy. Co je třeba si z této kapitoly odnést je, že RNN má vnitřní stav (paměť). Tato paměť se s každým vstupem aktualizuje a je použita pro výpočet dalšího výstupu.

Na obrázku 2 je znázorněna rekurentnost RNN a co tato rekurentnost v praxi znamená. Tedy že vstupem do rekurentní sítě není pouze vstupní vektor, ale i historie předchozích výstupů (tedy vnitřní stavy rekurentní sítě).

Jednoduchá RNN se v praxi příliš nepoužívá, protože zde dochází k postupnému mizení informací o příliš vzdálených vstupech (vzdálených myšleno objevili se o několik vstupů dříve).

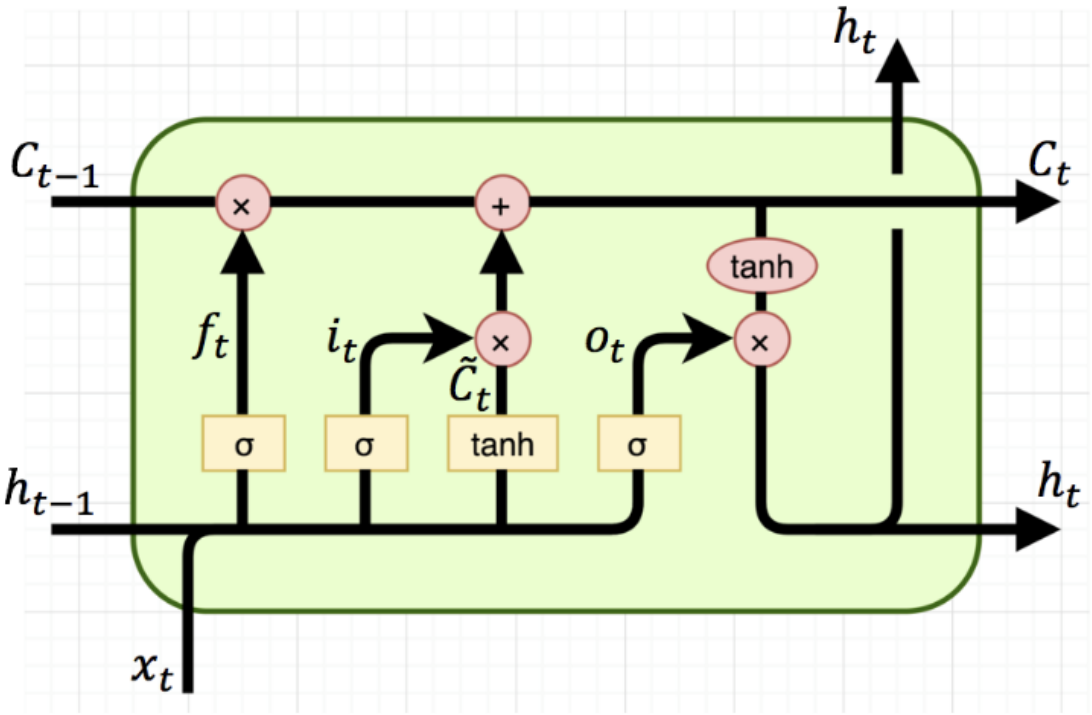
Například při predikci slov může být pro jednoduchou RNN předpovědět, že poslední slovo ve větě “Tento pes se jmenuje Alík a umí dobře štěkat.” bude právě slovo štěkat, což je způsobeno tím, že mezi slovem pes a štěkat je mnoho dalších slov které komplikují síti schopnost zapamatovat si, že věta pojednává o psovi. Tento problém řeší dále popsaná neuronová síť Long Short Term Memory.

3.6.2 Long Short Term Memory (LSTM)

Jak bylo řečeno u jednoduché RNN, LSTM [17] se snaží vyřešit neschopnost udržet si informaci o dávném vstupu v případě jednoduché RNN. Pro lepší představu je na obrázku 3 uvedena architektura jedné LSTM buňky. Tato buňka má poněkud složitější architekturu než jednoduchá RNN. Obsahuje takzvané brány a jsou hned tři (značené písmeny f,i,o). Každá z těchto bran je ohraničena funkcí sigmoid, která zajišťuje, že hodnoty se budou pohybovat v rozmezí 0 až 1. Brána f se v angličtině nazývá forget gate a pomáhá při rozhodování, které prvky z předchozího stavu mají být zachovány a které odstraněny. Brána i se v angličtině nazývá input gate neboli vstupní brána a stará se o výběr relevantních prvků ze současného vstupu. Poslední brána o se nazývá output gate neboli výstupní brána která, jak už název napovídá určuje, co bude výstupem dané buňky. O každé z bran se dá uvažovat jako o malé neuronové síti, která se učí, kdy se pracuje se kterými slovy.

Pro lepší pochopení bude uveden příklad. Předpokládáme datovou sadu, která obsahuje věty “Karel viděl Janu.”, “Jana viděla Kamila.” a “Kamil viděl Karla.” . Neberme teď v úvahu různé tvary jmen a fakt, že je to velmi malá datová sada. Po natrénování dostane síť slovo Karel. Síť si nejprve vygeneruje možná následující slova a tato slova pošle do vstupní brány. V této bráně se vyberou slova, která dávají větší smysl, že budou jako následující – síť se naučila, že po jménu je buď slovo viděl nebo konec věty, a tak vyloučí všechna jména z možného výstupu. Tato slova přicházejí do forget gate, kde se rozhoduje, jestli je možné nějaké slovo v paměti zapomenout. Předchozí slovo Karel je si ještě třeba pamatovat, a tak zůstane v paměti. Nakonec přichází zbylá slova do výstupní brány, kde se vybere to slovo, které má největší pravděpodobnost, že bude jako další a toto slovo je výstupem této iterace a vstupem do iterace další. Pokud je síť dobře naučena, bude výstupem a novým vstupem slovo viděl. U tohoto slova se síť naučila, že po něm následuje jméno, ale nikdy ne stejné jméno. Proto se ve vstupní bráně vyberou všechna ostatní možná jména a ta jsou poslána do forget gate. Zde se zjistí, že může být použito uložené jméno Karel. Proto dostane vyšší váhu jméno, které je s Karlem spojeno, což je v tomto případě Jana a jméno Karel může být odebráno z paměti. Ve výstupní bráně je jméno Jana jasným favoritem, a proto bude výstupem z neuronové sítě.

Na podobném principu funguje i Gated Recurrent Unit (GRU) [11], která má jednodušší architekturu (obsahuje o jednu bránu míň a historie je rovna výstupu v předchozím kroku), a proto není tak náročná na běh, i když nemusí vždy dosahovat tak dobrých výsledků jako LSTM.



Obrázek 3: LSTM Buňka (Christopher Olah 2015)

3.6.3 Obousměrná RNN

Obousměrná RNN [17] je běžná RNN, která nabízí lepší výkon v oblastech, kde je výhodné pro predikci počítat s daty jak z budoucnosti, tak z minulosti (budoucností jsou myšleny vstupy, které budou analyzovány po aktuálním vstupu). RNN jsou závislé na pořadí nebo času a určitá záměna tohoto pořadí může naprosto změnit to, co RNN z analyzovaných dat extrahuje. Této citelnosti na změnu pořadí využívá obousměrná RNN, která se skládá ze dvou normálních RNN vrstev. Podmínkou je, že obě vrstvy musí být stejné (pokud jeden směr používá LSTM vrstvu, pak i ten druhý musí využít LSTM vrstvu). Každá z těchto 2 vrstev zpracovává vstup v jednom směru (ve směru a v protisměru) a nakonec své výsledky sloučí. Tímto způsobem může obousměrná RNN poznat vzory, kterých by si běžné RNN vrstvy nevšimly. [10]

3.7 Konvoluční síť

Konvoluční síť [6] se stala velmi oblíbenou v rámci zpracování a klasifikace obrázků, pro její schopnost hledat a detekovat vzory ve vstupních datech. Avšak i textová data obsahují jisté vzory (například slova, která se často vyskytují blízko sebe), a proto se konvoluční síť dá využít i pro zpracování textových dat. Pro snazší pochopení bude uveden princip fungování sítě při zpracování obrázku a poté bude uveden rozdíl oproti zpracování textu.

Konvoluční síť se skládá z mnoha konvolučních vrstev, které provádí takzvanou konvoluci na vstupu. Kromě těchto vrstev obsahuje typicky vstupní vrstvu, výstupní vrstvu a mezi jednotlivými konvolučními vrstvami se můžou nacházet takzvané pooling vrstvy. Konvoluční vrstvy obsahují filtry, které jsou zodpovědné za samotnou detekci vzorů v obrázku. Zpočátku může jít o detekci hran, okrajů, kruhů a podobně. Tyto jednoduché tvary jsou poté využity hlouběji v sítí pro detekci složitějších objektů jako například dveře, okna a podobně. Pro každou vrstvu je specifikováno, kolik filtrů má obsahovat a o jaké velikosti. Filtry jsou čtvercové matice, jejichž hodnoty jsou zpočátku náhodné. Konvoluční vrstva aplikuje tento filtr na obrázek, a to tak, že ho postupně posouvá po obrázku a počítá skalární součin pro každý pixel obrázku, dokud nedojde na konec obrázku. Tomuto posunu po obrázku se říká konvolvování. Po konvolvování vznikne nová matice, která reprezentuje obrázek a která bude poslána hlouběji do sítě. Tato matice bude obsahovat vysoké hodnoty v místech, kde filtry našly vzory a velmi malé hodnoty v místech, kde hledané vzory nebyly. Po některých konvolučních vrstvách může následovat pooling vrstva, která slouží ke zmenšení velikosti zpracovávaného prvku, což snižuje paměťové nároky, zvyšuje rychlost sítě a zabraňuje přeučení. Pooling funguje tak, že se po obrázku posouvá okno o určitých rozměrech (například 2x2) a pro každé okno je získána hodnota, která bude součástí nového zmenšeného prvku. Na rozdíl od konvolučního filtru, kde se posouvaná okna překrývají, zde se okno posouvá tak, aby se nepřekrývala.

Při aplikování filtru může dojít k případu, že rozměr filtru není beze zbytku dělitelný rozměrem obrázku. V tomto případě se v určitých místech filtr nemůže zcela aplikovat a existují 2 strategie, jak tento problém řešit:

Same padding – Tento způsob aplikování filtru zajišťuje, že po konvoluci vznikne matice se stejným rozměrem, jako byla před konvolucí. Řeší se to tak, že okraje, kde už filtr přesahuje hranice obrázku, jsou vycpány 0.

Valid padding – Zde bude rozměr výsledné matice zmenšen tak, aby byl dělitelný velikostí filtru a to tak, že pixely, kde by filtr už přesahoval hranice obrázku, jsou ignorovány.

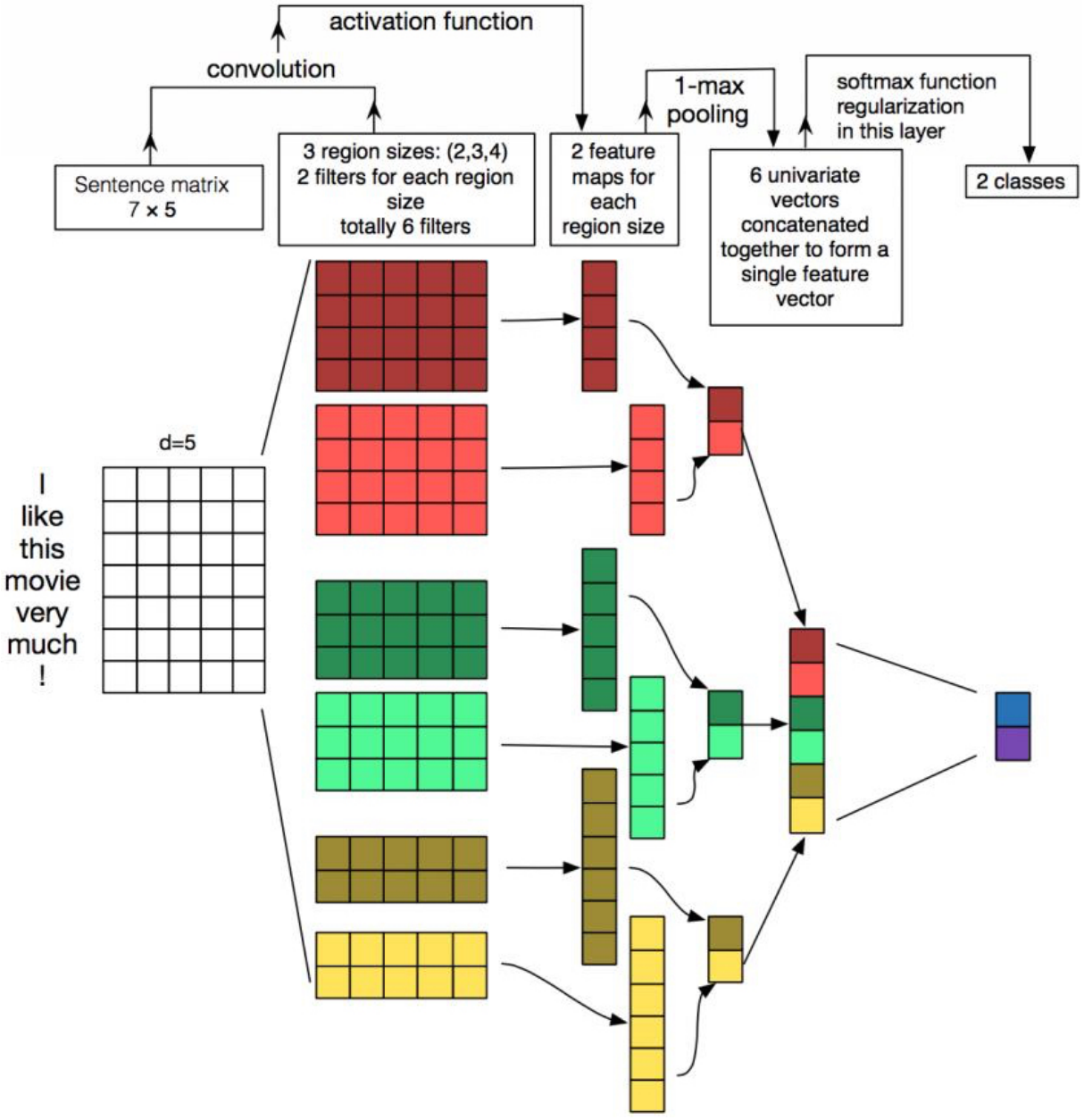
Při poolingu je zase třeba zvolit strategii, která bude použita pro získání nové hodnoty, kdy opět existují 2 způsoby:

Average pooling – V daném okně je vypočtena průměrná hodnota všech prvků, které se v aktuálním okně vyskytují.

Maximum pooling – Bere pouze maximální hodnotu daného okna.

Před výstupem je matice zploštěna do 1-D vektoru, aby mohla být vložena do hustě propojené vrstvy, ve které se počítá klasifikace.

V případě textu se každý filtr také chová jako detektor různých charakteristik. Pokud se charakteristika vyskytuje někde v textu, tak aplikace filtru na tuto oblast vrátí vysokou hodnotu, která bude po aplikaci max funkce zachována. Během trénování se síť sama naučí hodnoty svých



Obrázek 4: Konvoluční síť (Zhang, Y., & Wallace, B. (2015))

filtrů podle toho, jaký úkol má zrovna vykonávat. Při využití vnoření slov je na vstupu pro síť matice o rozměrech $n \times m$, kde n udává počet slov a m dimenzionalitu vnoření slov. Filtry posouvají po celých řádcích matice (po slovech). Tudíž šířka filtrů je většinou stejná jako šířka vstupní matice. Výška filtrů se může lišit, ale typická jsou posouvací okénka (v angličtině sliding windows) přes 2 až 5 slov. [17]

Na obrázku 4 je naznačeno fungování konvoluční sítě při klasifikaci. Na vstupu je matice o rozměrech 5×7. Síť obsahuje 6 filtrů, rozdělených do 3 skupin, kde každá skupina má rozdílnou velikost posouvacího okénka (tedy na kolik slov se bude najednou dívat). Šířka filtrů odpovídá šířce vstupní matice. Po aplikaci filtru na vstupní matici jsou vytvořeny mapy charakteristik z každého filtru, na které je aplikována 1-max sdružovací funkce, která z každé mapy získá jen největší hodnotu. Z těchto hodnot je poté vytvořen jeden vektor, na kterém je pak prováděna klasifikace. Je to pouze jednoduchá síť normálně může obsahovat více konvolučních a sdružovacích vrstev.

3.8 Vnoření slov

Vnoření slov (anglicky word embeddings) využívá distribuční hypotézy, podle které mají slova s podobným významem tendenci se vyskytovat v podobném kontextu. Hlavní výhodou je, že zachycuje podobnost mezi slovy. Vnoření slov se často používá jako první vrstva pro zpracování dat u hlubokých modelů. Typicky je vnoření slov předem natrénované na velkých neoznačkovaných korpusech, kde naučené slovní vektory mohou zachytit obecnou syntaktickou a sémantickou informaci, díky čemuž je tato struktura vhodná pro zachycení podobnosti kontextu, analogií a díky jejich menším dimenzionálním rozměrům jsou rychlé a účinné při základních výpočtech zpracování přirozeného jazyka. [5] [18]

Pro vytvoření vnoření slov jsou většinou využívány nehluboké neuronové sítě (shallow). Existuje více způsobů jak je vytvořit.

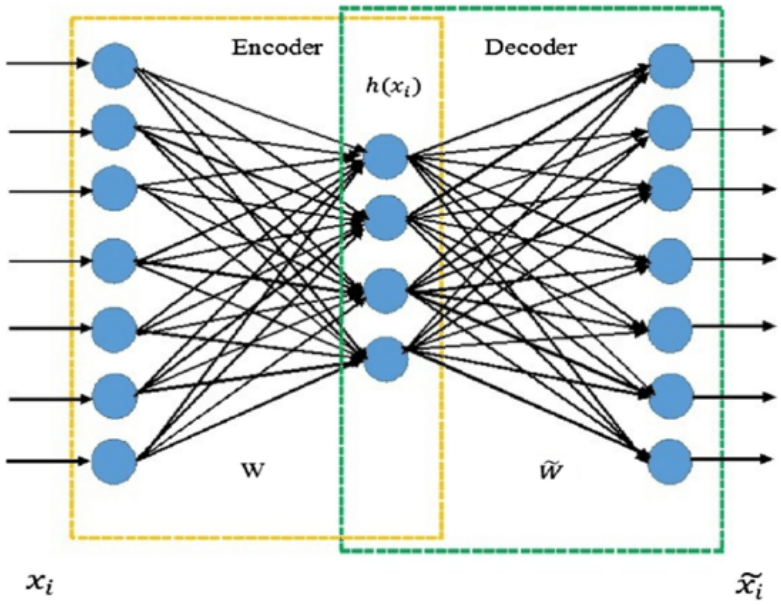
3.8.1 Vytvoření pomocí vrstvy vnoření

Vnoření slov je naučeno spolu s neuronovou sítí přímo na daný NLP problém. Vyžaduje předzpracované dokumenty v binárním kódování. Velikost vektorového prostoru je dána jako parametr a standardně bývá v dimenzích 50, 100 nebo 300. Tento přístup vyžaduje velké množství dat a může být pomalý, ale umožňuje vytvořit embedding přímo určené pro daná data a úkol.

3.8.2 Vytvoření pomocí Word2Vec

Word2Vec [18] je statistická metoda pro efektivní trénování samostatného vnoření slov z korpusu. Byly představeny dva odlišné modely, které mohou být využity jako část přístupu tohoto modelu pro naučení vnoření slov.

- Continuous Bag-of-Words (CBOW) - učí se vnoření tak, že předpovídá současné slovo na základě jeho kontextu.



Obrázek 5: Architektura autoenkodéru (Scientific Figure on ResearchGate 2018)

- skip-gram - učí se tak, že předpovídá okolní slova na základě daného slova.

Oba modely se zaměřují na naučení se informací o slově na základě jeho lokálního kontextu, kde tento kontext je definovaný okénkem sousedních slov, kdy toto okénko je nastavitelný parametr modelu. Výhodou tohoto modelu je, že se může efektivně (nízké nároky na paměť a čas) naučit velmi kvalitní vnoření slov, což v důsledku umožňuje naučit se větší vnoření slov z mnohem větších korpusů o rozsahu i miliardy slov.

3.9 Autoenkodér

Autoenkodér [9] je druh neuronové sítě, který se učí kopírovat vstup na výstup. Základní autoenkodér obsahuje jednu skrytou vrstvu, která by měla mít menší počet neuronů než vstupní vrstva. V opačném případě by se síť byla schopna naučit přesně kopírovat vstup na výstup. Úkolem této sítě je naučit se vstup zakódovat do reprezentace s menším počtem dimenzí tak, že jej poté bude schopna co nejpřesněji zrekonstruovat. Aby to dokázala, musí síť najít ty atributy vstupních dat, které je nejvíce charakterizují a tyto atributy uložit do vah. Autoenkodér se do jisté míry dá srovnat s metodou SVD, která taky vytváří reprezentaci dat v menší dimenzi. Na obrázku 5 je zobrazena základní architektura autoenkodéru.

Autoenkodéry mají širokou škálu využití. Jak už bylo zmíněno, mohou být využity pro zmenšení dimenzionality vstupu, dále se dá použít pro odstranění šumu v obrázcích, strojový překlad, sumarizaci textu nebo dokonce pro popisování obrázku.

3.10 TensorFlow a Keras

Všechny experimenty s neuronovými sítěmi v této práci jsou provedeny s pomocí Tensroflow a Keras.

TensorFlow [29] je otevřená knihovna pro číselné počty, která usnadňuje vytváření algoritmů strojového učení. Umožňuje vytvářet grafy datových toků, což jsou struktury, které popisují, jak se data pohybují skrze graf, což mimo jiné umožňuje vytvářet neuronové sítě. [33]

Keras [13] je vysokoúrovňové API pro vytváření a trénování neuronových modelů. Je zaměřeno na rychlý návrh, pro výzkum a také pro produkci. Hlavními výhodami jsou, že je uživatelsky přívětivé, a napsáno tak, aby bylo snadné jej použít pro běžné případy. Je snadné jej rozšířit o další vrstvy, metriky a ztrátové funkce. Toto API nemusí nutně pracovat s knihovnou TensorFlow, ale s verzí TensorFlow 2.0 je Keras součástí TensorFlow, což dělá použití s TensorFlow snazší než s ostaními knihovnami jako PyTorch.

4 Měřítka kvality modelů

V této krátké kapitole jsou pro úplnost popsány metody pro ověření míry kvality jednotlivých modelů, které byly při experimentech použity.

4.1 Klasifikace

Ověření přesnosti klasifikace se provádí porovnáním odhadované třídy a skutečné třídy. Výsledná přesnost modelu je pak vypočtena jako $\frac{\text{Počet správně klasifikovaných}}{\text{Celkový počet vzorku}}$.

Pro upřesnění, kde model nejčastěji chyboval, byla použita matice záměn, což je matice, která ve sloupcích obsahuje očekávanou hodnotu a v řádcích předpověď modelu. V ideálním případě má matice hodnoty větší než nula na diagonále a všude jinde jsou pak nuly. Pokud jsou mimo diagonálu nenulové hodnoty znamená to, že došlo k chybné klasifikaci a díky tomu se dá určit u jakých kategorií má zkoumaný model největší potíže.

4.2 Kvalita modelu topiků

Modelování topiků je o hledání hlavních témat, která se vyskytují ve velké a jinak nestrukturované kolekci dokumentů. [3] Výsledný model pak obsahuje slova tzv. topiky, rozdělené do jednotlivých témat. Avšak modely samy nezaručují, že nalezená slova jsou správná. Pro ověření kvality nalezených slov se používá koherence. Koherence je to, co dělá text významově smysluplný. Je to kontextový výskyt promluv v textu, tedy to, jak dobře jednotlivá slova kontextově zapadají do textu a jak moc přispívají k pochopení významu nebo zprávy. [7] Existuje více metod pro měření koherence. Knihovna Gensim [24] tyto míry implementovala na základě publikace [23]. Metody použité v této práci jsou UCI a UMASS.

4.2.1 Koherence UCI

Tato koherence měla největší korelaci s hodnoceními provedeným člověkem. Je založena na klouzavém okénku a bodově vzájemné informaci (anglicky pointwise mutual information) všech párů slov z poskytnutých slov.

4.2.2 Koherence UMass

Tato koherence je založena na myšlence, že výskyt každého důležitého slova by měl být podporován každým předcházejícím důležitým slovem. Tedy pravděpodobnost, že se důležité slovo objeví, by měla být vyšší, pokud dokument už obsahuje důležitější slovo stejného tématu.

5 Datové sady

Pro testování byly použity datové sady Reuters, 20 Newsgroups, DBpedia Ontology Classification Dataset, AG's News Topic Classification Dataset a Yelp. Byla také vytvořena jedna česká datová sada, využívající popisy filmů z ČSFD. Tyto datové sady se různě liší svou velikostí, původem (články, recenze, diskuse), počtem a rozložením témat. Každá datová sada byla před samotnými testy různě upravena. Provedené úpravy jsou popsány v následujícím výčtu.

5.1 Reuters

[16] Datová sada obsahující přes 20000 článků od agentury Reuters. Obsahuje také přes 60 témat. Témata mají ale nevyvážené rozložení, kdy u některých existuje více jak 100 článků, ale některé obsahují například jenom jeden článek. Pro lepší výsledky testů byla proto datová sada upravena tak, aby obsahovala pouze témata s články, kterých je více jak 100. Tato úprava byla provedena na trénovací části datové sady. V testovací části pak byly všechny články, jejichž témata se nevyskytovala v trénovací části, odstraněny. Ve výsledku datová sada obsahuje 5770 trénovacích dokumentů, 2255 testovacích dokumentů a dokumenty jsou rozděleny pouze do 10 témat.

5.2 20 Newsgroups

[19] Datová sada, obsahující emailovou korespondenci rozdělenou do 20 skupin. Pro získání této sady byla použita knihovna Scikit-learn [27], která danou datovou sadu obsahuje spolu s metodami, které umožnily extrahovat pouze určité části (například pouze tělo zprávy). Problém této sady byl, že některé emaily byly velmi krátké (méně jak 10 slov) občas i prázdné a po odstranění stopslov se tento počet ještě zvýšil. Aby nezhoršovaly výsledky testování, byly tyto krátké zprávy odstraněny jak z trénovací, tak z testovací části. Po redukci obsahovala sada 10332 trénovacích a 6887 testovacích článků.

5.3 DBpedia Ontology Classification Dataset

[35] Datová sada sestavena výběrem z 14 nepřekrývajících se témat z DBpedia 2014. Každé téma má 40000 trénovacích a 5000 testovacích článků, což dohromady vytváří datovou sadu o 560000 trénovacích článcích. Průměrná délka článku je 46 slov. Z toho důvodu byla sada upravena tak, že články s délkou menší než 46 slov byly z datové sady odstraněny. Po této úpravě se výsledný průměrný počet slov zvýšil na 76. Další úpravou bylo odstranění těch témat, která měla méně jak 5000 článků pro zachování rovnoměrné distribuce jednotlivých témat. Po veškerých úpravách zůstalo 11 témat, kde nejméně zastoupené téma bylo NaturalPlace s 5479 články a nejvíce zastoupené bylo téma Building s 9470 články. Výsledný počet článků, rozdělených do 11 témat, byl pro trénovací část 80259 a 10092 pro testovací část.

5.4 AG's News Topic Classification Dataset

[35] Datová sada obsahující 120000 článků z více jak 2000 zdrojů rozdělených do 4 témat, kde každé téma má 30000 článků. Průměrná délka článku je 31 slov. Výsledná testovací sada byla upraven podobně jako předchozí s rozdílným limitem odstranění, tedy články s délkou menší než 31 slov byly z datové sady odstraněny, čímž se průměrný počet slov zvýšil na 58. Výsledný počet trénovacích článků byl pouhých 4954. Distribuce do témat byla (1393, 1134, 798, 1628). Testovacích článků bylo 302.

5.5 Yelp

[34] Datová sada obsahující přes 6 milionu recenzí. Kategorie jsou rozděleny podle jednotlivých institutů, na které byly psány recenze a ve většině případů obsahují více jak jedno klíčové slovo. Na takto rozsáhlá jména kategorií by se obtížně dělaly testy, a proto byly recenze rozděleny jenom podle jednoho klíčového slova. Například pokud měla kategorie 10 slov a 1 z těchto slov bylo klíčové, byla celá kategorie zařazena do nové kategorie, která byla pojmenována podle daného klíčového slova. Rozdělování probíhalo tak, že se zjistilo klíčové slovo s největším odhadovaným zastoupením, podle tohoto slova byla sada rozdělena na dvě menší, kdy jedna obsahovala všechny recenze s kategorii daného klíčového slova a druhá obsahovala články, které dané slovo v kategorii neobsahovaly. Tento zbytek byl rozdělen na další dvě části podle dalšího, nejčastěji zastoupeného slova. Rozdělování bylo prováděno tak dlouho, dokud vznikaly dostatečně obsáhlé sady s danou kategorií. Limit byl stanoven na velikost 30 MB. Kategorie, které měly více jak 10 slov byly z datové sady vyřazeny, aby bylo zajištěno, že recenze nebudou mít příliš široké zaměření. Konečná sada obsahovala 8 témat a měla dohromady velikost 2,24 GB. V rámci testování byl z časových a paměťových nároků vybrán pouze zlomek z celkové sady (pro každé téma bylo vybráno 10000 recenzí pro trénování a 1250 pro testování). Výběr byl prováděn náhodně. Na rozdíl od ostatních sad zde nebyly odstraněny příliš krátké recenze.

5.6 Česko-Slovenská filmová databáze - ČSFD

[8] Česká datová sada, která byla vytvořena stažením popisů filmů a jejich kategorií z webu Česko-Slovenské filmové databáze pomocí upravené knihovny csfd-parser [12]. Webová stránka umožňuje hledání filmů podle id a tak byly popisy stahovány postupným procházením jednotlivých id filmů. Často filmy neobsahovaly popisy, popisy byly velmi krátké, nebo film s daným id vůbec neexistoval a tak z 555000 prozkoumaných id bylo získáno pouze necelých 132000 filmových popisů. Stejně, jako v případě datové sady Yelp, i zde bylo více kategorií u jednotlivých filmů, kde valná většina měla maximálně 4 popisná slova.

Pro vytvoření sady, která by se dala použít pro klasifikaci bylo vyzkoušeno více způsobů, kde každý má své výhody a nevýhody. Před samotným vytvořením sady bylo několik slov v kategoriích sloučeno do jednoho (například slova thriller a krimi byla sloučena do slova akční).

Dále byla některá slova, která neměla informativní hodnotu odstraněna (slova animované a krátkometrážní).

První způsob se z části podobal způsobu vytváření datové sady Yelp a orientoval se na jednotlivá kategorická slova. Nejdříve byla zjištěna četnost těchto slov, a podle počtu požadovaných témat bylo vybráno N nejčetnějších slov. Ostatní slova byla z kategorických popisků odstraněna. Poté byly postupně popisy filmů řazeny k nalezeným slovům s tím, že se preferovaly kategorie těch nejméně četných slov, aby se nestalo, že nejčetnější kategorie by pohltila část méně četné kategorie. Dále také byl pro každé téma vybírán stejný počet popisů, kde počet odpovídal počtu popisů u nejméně četného tématu. Tato strategie byla použita za účelem vyváženosti datové sady. Nakonec byla roztříděná témata zamíchána a poté rozdělena v poměru 80 % pro trénovací a 20 % pro testovací část. Tento přístup měl nevýhodu v tom, že určité žánry se skládají ze 2 slov jako například akční komedie. Použitím jednoslovných kategorií způsobilo, že klasifikátory často špatně klasifikovaly filmy, kterých se tento 2 slovný popis týkal.

Druhý způsob se pokoušel řešit problém 2 a více slovných kategorií. Nejdříve odstraní méně četná kategoriální slova stejně jako při prvním způsobu a to proto, že tato slova většinou nebyla velmi informativní a jenom snižovala výsledný počet popisů, ze kterých bylo možné vybírat. Po odstranění málo četných slov z kategorií byly kategorie opět spočítány, tentokrát se ale brala v potaz všechna slova. Počet témat byl nastavitelný parametr a v rámci testů bylo jako počet témat stanoveno číslo 10. Dále je postup totožný s prvním způsobem s tím rozdílem, že se v kategorii musí vyskytovat jenom daná kategoriální slova. I zde bylo zavedeno omezení v počtu popisů v jednotlivých tématech pro zajištění vyváženosti datové sady. Pokud se vyváženost nekontrolovala vznikaly datové sady, kdy rozdíl mezi nejméně a nejvíce četnými tématy byl i desetinásobně větší.

Třetí způsob je spíše doplňkem ke zbylým dvěma, kdy z kategoriálních slov bylo odstraněno nejčastější slovo drama, které často nepomáhalo témata rozlišit a jenom snižovalo počet popisů pro jednotlivá témata.

6 Klasické modely - experimenty

Experimenty se zaměřovaly hlavně na zjištění přesnosti algoritmů, popsaných v sekci 2. Prováděly se na datových sadách, zmíněných v sekci 5. Nejprve byly testy zaměřeny na nalezení optimálních parametrů jednotlivých modelů. Po nalezení parametrů se testy zaměřily na optimální způsob předzpracování. Nakonec byly modely testovány s daným optimálním nastavením.

6.1 Postup testování

Při testování byla největší pozornost věnována vlivu předzpracování (různé kombinace pro odebrání stopslov, čísel, použití stemmeru, použití lematizátoru a odebrání krátkých slov) na přesnost. Další oblastí bylo testování vlivu nastavených parametrů u jednotlivých modelů, ale výsledky těchto testů zde nejsou uvedeny, pouze je zmíněno, jaké nastavení se ukázalo jako nejoptimálnější. Testy modelů s optimálně nastavenými parametry byly prováděny na jednotlivých způsobech předzpracování a pokud model využíval náhodu, bylo těchto testů provedeno vždy alespoň 5. Po nalezení optimálního způsobu předzpracování byly testy prováděny pouze na tomto optimálním způsobu. Kroky v testech:

1. Načtení datové sady s daným způsobem předzpracování.
2. Vytrénování modelu na trénovací části datové sady a následné testování modelu.
3. Vytvoření matice záměn a vypočtení přesnosti daného nastavení (kolikrát se model 'trefil' do správného tématu).

Témata jednotlivých dokumentů, které modely odhadovaly, byla hodnocena různě podle způsobu implementace daných modelů a reprezentace jejich výsledků.

6.1.1 Testování modelu LDA

Testování modelu LDA nebylo úplně triviální. Po natrénování totiž nebylo snadné určit, které téma patří kterému indexu v natrénovaném modelu. Proto se před samotným testováním ještě hledal odhad, ke kterému indexu které téma patří. Bylo experimentováno se dvěma způsoby.

První způsob fungoval tak, že se vždy vzaly z trénovací části všechny články jednoho tématu a ty se následně aplikovaly na natrénovaný model. Index, který se v odhadu nejčastěji vyskytoval, byl přidělen testovacímu tématu. Často se stávalo, že ne všechna témata měla přidělen index - pro více témat vyšel nejlíp stejný index. V tomto případě se předpokládalo, že jsou si daná témata podobná a index reprezentoval obě témata.

Nakonec bylo třeba vyřešit, co dělat v případě, kdy se nepřiřazený index objeví u testování. V tomto případě byl tématu tento nepřiřazený index přidělen a daný článek nebyl testován. Zde se předpokládá, že pokud daný index skutečně reprezentuje dané téma, bude se pro toto téma opakovat. Pokud tomu tak není, je to špatně naučený index, který snižuje přesnost modelu.

Tento způsob testování mohl mít uměle vysokou přesnost v případě, že bylo přiděleno méně jak 50 % indexů před testováním. Mohl nastat případ, že 3 indexy reprezentovaly 10 témat, což snížilo šance, že se model splete a pokud se spletl, pořád měl šanci, že index přidělený až při testování bude odpovídat testovanému tématu.

Z důvodu problému uměle vysoké přesnosti byl vytvořen druhý způsob. Ze začátku funguje stejně jako první metoda, ale jednou přiřazený index byl u dalších témat ignorován. Takže pokud se stalo, že u dvou témat vyšel nejlépe stejný index, byl v druhém případě tento index ignorován a jako index tématu byl vybrán první, který doposud nebyl přiřazen. Tímto bylo zajištěno, že všechna témata měla přiřazen index nebo alespoň dva různé indexy neidentifikovaly stejné téma. Díky tomu, že tento způsob eliminoval problém uměle vysoké přesnosti, byl nakonec použit i při testování přesnosti modelu LDA.

6.1.2 Testování Naive Bayes

V případě Naive Bayes bylo testování snadné, neboť model pro testovaný článek vracel jak číslo odhadovaného tématu, tak i procentuální pravděpodobnost, s jakou do daného tématu patří. Vzhledem k tomu, že výsledky testů tohoto modelu byly vždy shodné, byl tento model testován pouze jednou pro dané předzpracování dokumentu na rozdíl od LDA, kde bylo prováděno alespoň 5 testů na jedno předzpracování.

6.1.3 Testování SVM

SVM bylo testováno stejně jako Naive Bayes, protože obě implementace pocházely ze stejných knihoven a tedy i SVM pro každý testovaný článek vracel číslo odhadovaného tématu a procentuální jistotu, že do daného tématu patří. Obdobně byl SVM testován pouze jednou pro dané předzpracování.

6.1.4 Testování LSA

V případě LSA byly vyzkoušeny 2 způsoby testování. První byl stejný jako při testování LDA z knihovny Gensim (měly podobný výstup a proto šel aplikovat stejný způsob testu). Zde však nastal zmíněný problém s uměle vysokou přesností z důvodu nízkého počtu přidělených indexů před testováním. Počet přidělených indexů byl ještě menší než v případě LDA, a proto byl nahrazen druhým způsobem, který využívá matici podobnosti, která jako vstup bere korpus transformovaný do LSA prostoru. Při hledání tématu se podle podobnosti najde nejpodobnější článek v matici podobnosti a číslo tématu tohoto článku je vráceno jako odpověď (témata dokumentů jsou uložena v samostatném poli, protože matice podobnosti obsahuje pouze samotné dokumenty). Použití matice podobnosti je popsáno na stránkách knihovny Gensim [25]. Použití matice podobnosti způsobilo, že výsledky byly podobně jako u NB vždy shodné a proto i zde byl prováděn pouze jeden test.

6.1.5 Dobré parametry

U LDA se ukázalo, že největší vliv na přesnost měl počet iterací a průchodů korpusem. U LSA bylo lepší model trénovat na korpusu, který byl zpracován pomocí TF-IDF. Dále bylo experimentováno s počtem témat, kde se ukázalo, že udávat přesný počet témat vrací lepší výsledky.

Model Naive Bayes měl jeden nastavitelný parametr alfa, kde se ukázalo, že hodnota 0,1 je velmi robustní a podávala dobré výsledky na všech zkoumaných datových sadách. SVM měl nastavitelné parametry C (nastaven na 100), druh kernelu (zvolen rbf) a gamu (nastavena na 1). U rozhodovacího stromu přesnost nejvíce ovlivňoval počet charakteristik, které si měl strom pamatovat. Tento parametr byl nastaven na 10000. Nakonec u náhodného lesu byl kromě počtu charakteristik důležitý počet estimátorů, který byl stanoven na 20. Parametry, které zde nejsou zmíněny, měly minimální vliv na přesnost a byly ponechány hodnoty stanovené vývojáři jednotlivých modelů.

6.2 Vliv způsobu předzpracování na přesnost modelu

Jak již bylo řečeno, před samotným testováním se nejdříve hledal optimální způsob předzpracování. Po několika spuštěných testech se zdálo, že předzpracování má malý vliv na přesnost jednotlivých modelů. Proto byly spuštěny testy předzpracování na všechny datové sady. Testy na všech datových sadách byly prováděny pouze s modelem Naive Bayes. Testovaly se kombinace pěti technik předzpracování, a to odebrání čísel, použití stemmeru, použití lemmatizátoru, odebrání krátkých slov, a nakonec odebrání stopslov, dohromady tedy 32 testů na datovou sadu. Výsledky překvapivě potvrdily, že předzpracování ve většině případů mělo malý vliv. Jedinou výjimkou byla sada 20 Newsgroups, kde byl rozdíl mezi nejhorší a nejlepší přesností 10 %. Tento rozdíl většinou způsobovalo neodebrání stopslov (při ponechání stopslov byl výsledek horší), což mohla způsobit dopisová forma dané sady.

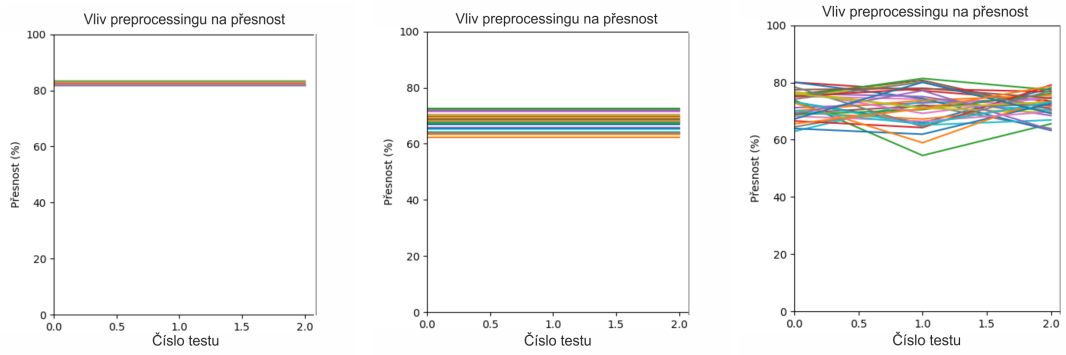
U datové sady Reuters se přesnost nejlepšího a nejhoršího předzpracování lišila o 5 %. U zbylých dvou sad byl rozdíl do 2 %. Tyto nízké rozdíly nejspíše způsobila opět struktura datové sady, kdy se jedná o články, ve kterých se nevyskytuje tolik stopslov jako v případě korespondence mezi uživateli fóra.

Tabulka 1: Nejlepší a nejhorší přesnost Naive Bayes při různých metodách předzpracování

Model	Nejhorší	Nejlepší
Reuters	77,61 %	83,02 %
AG's News	81,79 %	83,44 %
DBpedia	91,31 %	92,82 %
20 Newsgroups	59,93 %	70,35 %
Yelp	89,79 %	90,37 %

6.2.1 Vliv předzpracování podle modelu

Vliv předzpracování podle modelu byl testován hlavně na datové sadě AG's News. Na různé předzpracování reagoval každý model různě. Nejmenší vliv byl zaznamenán u Naive Bayes, kde se rozdíly v přesnosti pohybovaly v rozmezí 3 %. U LSA již šel vliv předzpracování poznat a přesnost se pohybovala v rozmezí 15 %. LDA od Sklearn na tom bylo obdobně, kde se přesnost pohybovala v rozmezí 15 %.



(a) Naive Bayes (b) LSA (c) LDA-Sklearn

Obrázek 6: Vliv předzpracování podle modelu u datové sady AGs News

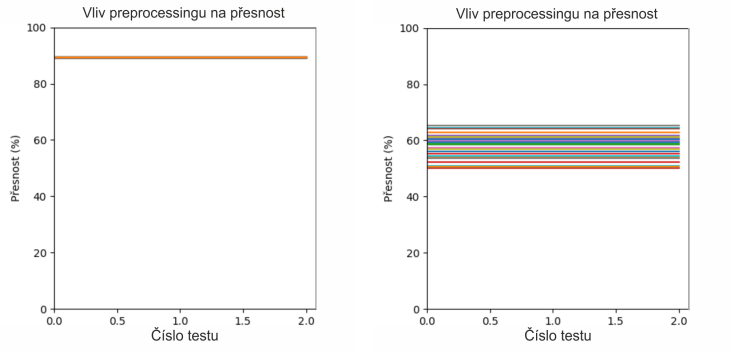
6.2.2 Vliv odstranění krátkých článků

Poslední testovanou oblastí byl vliv odstranění článků, které byly kratší než průměrná délka článku v původní datové sadě, (zmíněno v kapitole 5). Zde nebyl z časové náročnosti testován model LDA. Naive Bayes měl překvapivě lepší přesnost pro datovou sadu, ze které nebyly krátké články odstraněny, také rozmezí přesnosti se nepatrně zlepšilo. V případě LSA už neodstranění krátkých článků mělo negativní vliv jak na přesnost, tak na rozmezí, v jakém se přesnosti pohybovaly. [3]

6.3 Výsledky testů

Byly testovány 2 implementace modelu LDA (Gensim a Sklearn), model LSA (Gensim), Naivní Bayes(Sklearn), SVM (Sklearn), rozhodující strom (Sklearn) a náhodný les (Sklearn). Pro modely LDA bylo spuštěno 5 testů a výsledná přesnost byla určena z průměrů jednotlivých výsledků. Výsledky ukazují 2 přesnosti, a to s předzpracováním (kdy byly odstraněna stop slova, interpunkce a mnohočetné mezery).

[3] Statistiky v obrázcích jsou pouze pro původní sady. Pro sady, kde byly odebrány krátké články, jsou statistiky v obrázcích zobrazeny v kapitole 6.2.1



(a) Naive Bayes (b) LSA

Obrázek 7: Vliv odstranění krátkých článků na přesnost

6.3.1 Reuters

Pro datovou sadu Reuters měl největší přesnost Naivní bayes s 95,3 % při použití předzpracování. Některé modely ale lépe pracovaly s nepředzpracovanou verzí, kdy druhá nejlepší přesnost patří SVM, u kterého nebylo použito předzpracování. Nejhůře si vedl model LDA-Sklearn, který dosáhl 62,88 % přesnosti při předzpracování a 49,85 % bez použití předzpracování.

Tabulka 2: Přesnost jednotlivých modelů pro datovou sadu Reuters

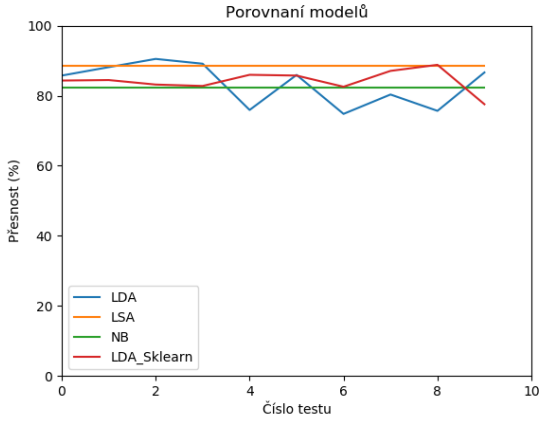
Model	Přesnost (předzpracování)	Přesnost (bez předzpracování)
LDA-Sklearn	62,88 %	49,85 %
LDA	65,10 %	62,39 %
LSA	86,52 %	83,10 %
NB	**95,30 %**	94,06 %
SVM	94,72 %	94,99 %
DT	89,80 %	90,78 %
RF	92,37 %	93,44 %

6.3.2 AG's News

U této datové sady si nejlépe vedl SVM s 86 % přesností při použití předzpracování. Druhý nejlepší model byl Naivní Bayes tentokrát bez použití předzpracování s přesností 85,76 %. Nejméně přesný byl model LDA-Sklearn s 41,64 % při použití předzpracování.

6.3.3 DBpedia

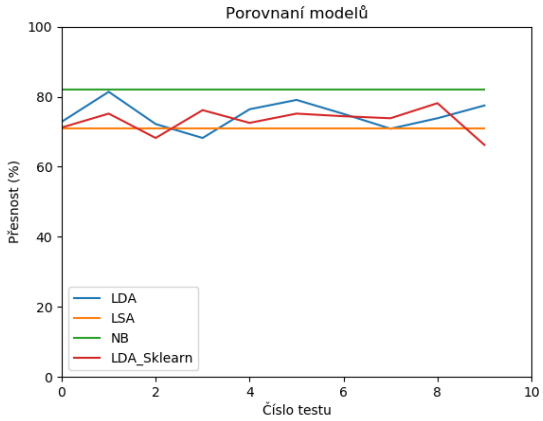
Stejně jako u sady AG's News i zde byl nejpřesnější model SVM s přesností téměř 98 % bez použití předzpracování. O 4 % horší byl druhý nejpřesnější model Naivní Bayes s přesností 93,52 % opět bez použití předzpracování. Obdobně dobře si vedl i náhodný les. Nejhůře si vedl model LDA z knihovny Gensim, jehož přesnost byla pouhých 43,78 % opět bez použití



Obrázek 8: Naměřené přesnosti jednotlivých modelů pro datovou sadu Reuters

Tabulka 3: Přesnost jednotlivých modelů pro datovou sadu AGNews

Model	S předzpracováním	Bez předzpracování
LDA-Sklearn	41,64 %	42,55 %
LDA	65,48 %	49,55 %
LSA	68,21 %	68,87 %
NB	83,44 %	85,76 %
SVM	**86,75 %**	86,09 %
DT	72,52 %	68,54 %
RF	83,77 %	81,13 %

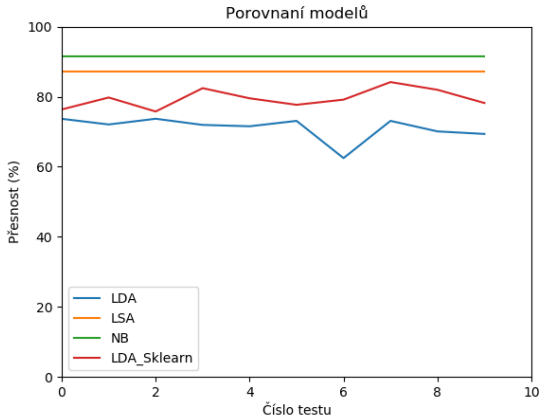


Obrázek 9: Naměřené přesnosti jednotlivých modelů pro datovou sadu AG's News

předzpracování. Velmi vysoká přesnost nejpřesnějších modelů na této sadě může být způsobena velkým množstvím článků, na kterých se mohly modely učit.

Tabulka 4: Přesnost jednotlivých modelů pro datovou sadu DBpedia

Model	S předzpracováním	Bez předzpracování
LDA-Sklearn	66,72 %	66,78 %
LDA	70,95 %	43,78 %
LSA	86,51 %	85,44 %
NB	92,97 %	93,52 %
SVM	96,99 %	**97,59 %**
DT	89,57 %	89,00 %
RF	92,43 %	92,45 %



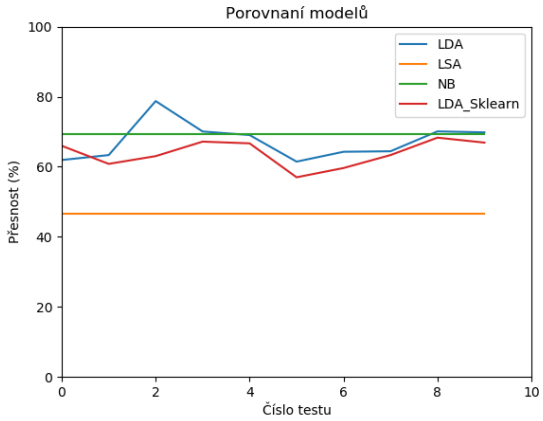
Obrázek 10: Naměřené přesnosti jednotlivých modelů pro datovou sadu DBpedia

6.3.4 20 Newsgroups

Zde byl nejpřesnější NB s 72 % bez použití předzpracování. Nejhůře si vedl model LDA, který měl přesnost pouhých 7,2 %. Modely měly při této sadě v průměru nejhorší přesnost, což nejspíše způsobuje fakt, že sada obsahuje zprávy mezi uživateli, kde tyto zprávy často obsahují překlepy, útržky zdrojových kódů, nebo adresy pro soubory na disku, které nebudou předzpracováním použitým pro toto testování odstraněny a pouze přispějí ke zvýšené nepřesnosti. Dalším faktorem mohl být velký počet témat (20), z čehož některé jsou reprezentovány jako podskupiny (například věda se dále dělí na medicínu, vesmír atp.). Na zmíněné problémové dvě vlastnosti datové sady byl proveden specifický test, který se snažil dopad vlastností na přesnost zmenšit. Nejprve byl zmenšen počet témat na 7, kde zmíněné podskupiny byly sloučeny do jedné nadskupiny. Touto redukcí se přesnost zlepšila o průměrně 5 %. Dále byl odstraněn html kód (který mohl být součástí samotné zprávy), text mezi hranatými závorkami, zkrácené výrazy byly převedeny na jejich plné ekvivalenty (don't na do not) a také byly odstraněny non-ASCII znaky. Po odstranění

Tabulka 5: Přesnost jednotlivých modelů pro datovou sadu 20 Newsgroups

Model	S předzpracováním	Bez předzpracování
LDA-Sklearn	8,79 %	7,78 %
LDA	16,36 %	7,20 %
LSA	44,52 %	21,35 %
NB	72,28 %	**72,43 %**
SVM	70,11 %	68,95 %
DT	44,62 %	41,27 %
RF	53,23 %	51,70 %



Obrázek 11: Naměřené přesnosti jednotlivých modelů pro datovou sadu 20 Newsgroups

zmíněných prvků byla přesnost překvapivě horší asi o 2 % (navíc předzpracování trvalo asi 3krát déle). Důvodem nižší přesnosti by mohlo být, že odstraněné prvky, leč se dají označit za šum, jsou specifické pro daná témata, a tudíž ve výsledku přesnost spíše zlepšily (například zdrojový kód se určitě nebude vyskytovat ve zprávách o motocyklech).

Tabulka 6: Přesnost jednotlivých modelů pro datovou sadu 20 Newsgroups se 7 tématy

Model	Přesnost
LDA	71,47 %
LSA	54,34 %
LDA-Sklearn	80,95 %
Naive Bayes	72,17 %

6.3.5 Yelp

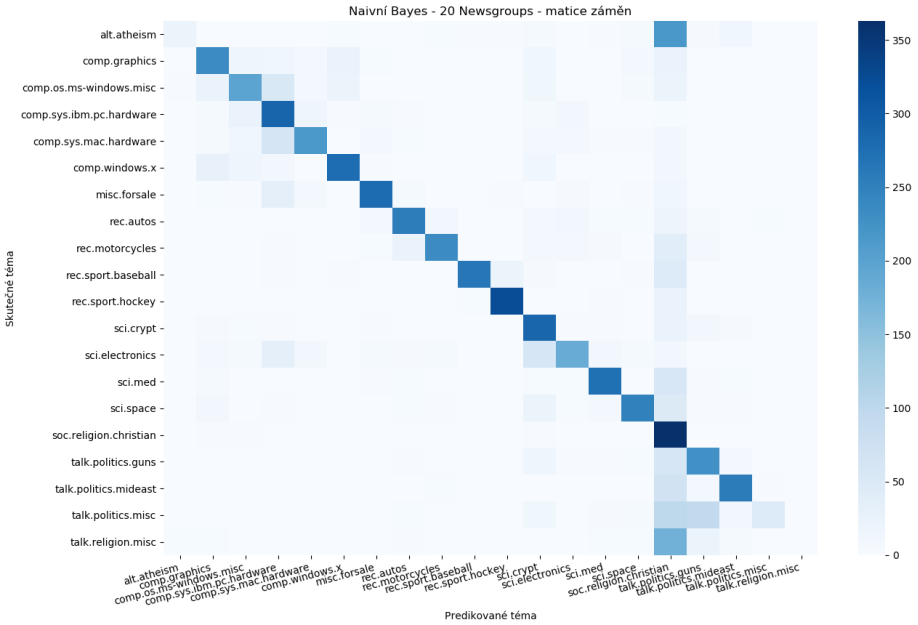
Vzhledem k tomu, že Yelp se skládá pouze z recenzí, dalo by se předpokládat, že výsledky budou podobné sadě 20 Newsgroups. Ale jak již bylo zmíněno - 20 Newsgroups obsahuje 20 témat, kdy některá jsou si částečně podobná, a rozdělení zpráv do jednotlivých témat není rovnoměrné. Na druhou stranu Yelp, který byl testován, byl vytvořen s rovnoměrným rozdělením, a navíc měl jenom 8 témat, což znamenalo méně možností pro model, aby se spletl. Tyto dva faktory nejspíše způsobily, že SVM dosáhl přesnosti 93 % když nebylo použito předzpracování, ale i při použití předzpracování byla přesnost téměř totožná s rozdílem pouhých 0,06 %. I ostatní modely si vedly lépe, než v případě sady 20 Newsgroups.

Tabulka 7: Přesnost jednotlivých modelů pro datovou sadu Yelp

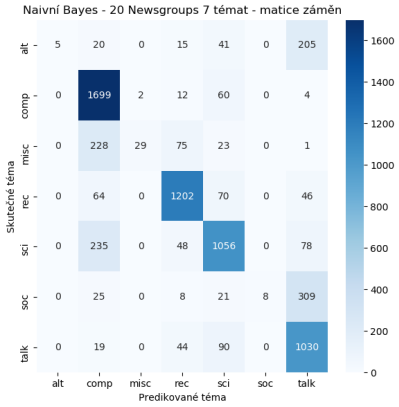
Model	S předzpracováním	Bez předzpracování
LDA-Sklearn	70,39 %	66,39 %
LDA	51,93 %	45,28
LSA	68,26 %	44,86 %
NB	90,83 %	90,65 %
SVM	92,94 %	**93,00 %**
DT	78,31 %	76,13 %
RF	83,62 %	82,62 %

6.3.6 ČSFD

V rámci vytváření této datové sady bylo vyzkoušeno více způsobů generace, což bylo zmíněno v kapitole 5.6. Výsledky ukazují přesnost modelů na čtyřech datových sadách, vygenerovaných ze stažených filmových popisů. Jedná se o jednoslovné popisy, víceslovné popisy s nevyváženým počtem jednotlivých kategorií, víceslovné popisy s vyváženým počtem v jednotlivých kategoriích, a nakonec datová sada obsahující vyvážené víceslovné kategorie s vynechanou kategorií drama, která byla odstraněna z důvodu velmi častého výskytu.

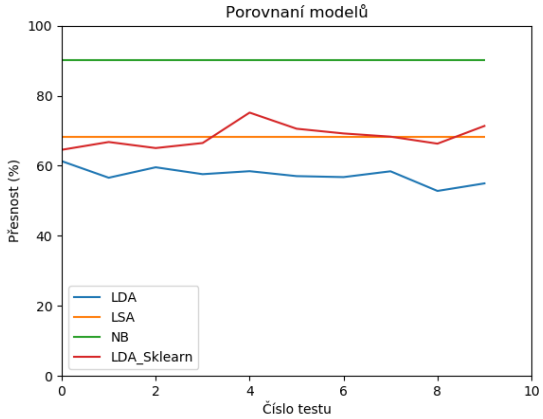


(a) 20 témat



(b) 7 témat

Obrázek 12: Matice záměn pro model Naive Bayes



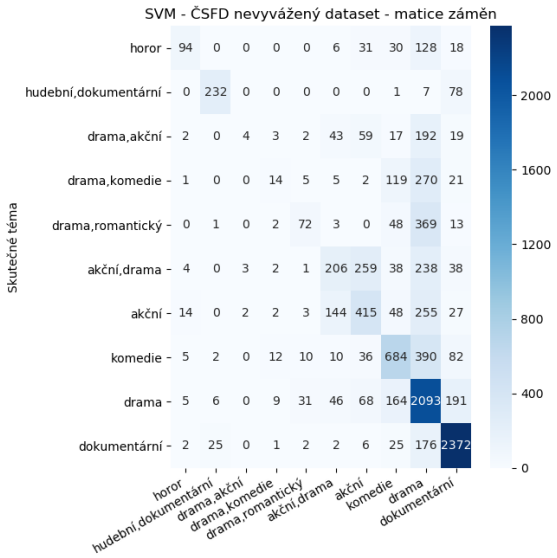
Obrázek 13: Naměřené přesnosti jednotlivých modelů pro datovou sadu Yelp

Při testech byly nejlepší výsledky dosaženy v nevyvážené datové sadě s víceslovnými kategoriemi. Avšak tato dobrá přesnost je způsobena nevyváženou testovací sadou, kdy model určil dobře nejčastěji se vyskytující kategorii viz. obrázek 14.

V případě vyvážených datových sad se vliv sestavení lišil od jednotlivých modelů. V případě nejpřesnějších modelů SVM a Naivního Bayesu, kde se přesnost pohybovala okolo 50 %, se nejvíce osvědčilo použití víceslovných kategorií a odstranění slova drama z těchto kategorií. Ostatní modely lépe reagovaly na víceslovnou datovou sadu, kde bylo slovo drama ponecháno. I použití předzpracování mělo různý vliv na přesnost podle modelu. U nejpřesnějších modelů přesnost spíše snižovalo, avšak rozdíly se pohybovaly okolo 1 %. V případě méně přesných modelů se přesnost zlepšovala a také rozdíly byly znatelnější (v určitých případech byla přesnost lepší i o 6 %.).

Tabulka 8: Přesnost jednotlivých modelů pro datovou sadu ČSFD (bez předzpracování)

Model	Jednoslovní	Nevyvážený	Vyvážený	Vyvážený (bez drama)
LDA-Sklearn	15,44 %	24,09 %	15,52 %	12,27 %
LDA	13,30 %	18,17 %	15,79 %	13,84 %
LSA	18,69 %	30,98 %	22,70 %	18,35 %
NB	48,29 %	58,18 %	49,00 %	50,16 %
SVM	**48,51 %**	**61,46 %**	**50,19 %**	**51,63 %**
DT	26,90 %	38,91 %	30,14 %	26,21 %
RF	36,55 %	49,50 %	37,18 %	35,50 %



Obrázek 14: Matice záměn pro nevyváženou datovou sadu ČSFD (pro model SVM)

Tabulka 9: Přesnost jednotlivých modelů pro datovou sadu ČSFD (s předzpracováním)

Model	Jednoslovní	Nevyvážený	Vyvážený	Vyvážený (bez drama)
LDA-Sklearn	15,43 %	24,45 %	16,31 %	13,51 %
LDA	16,28 %	26,73 %	19,21 %	14,29 %
LSA	23,34 %	37,61 %	29,58 %	24,34 %
NB	48,48 %	58,65 %	49,44 %	50,16 %
SVM	**48,65 %**	**61,45 %**	**50,25 %**	**50,82 %**
DT	28,27 %	39,96 %	32,11 %	28,19 %
RF	37,50 %	49,53 %	38,02 %	35,96 %

7 Neuronové sítě - experimenty

Podobně jako v předchozí sekci i u neuronových sítí se testy zaměřovaly na otestování přesnosti jednotlivých modelů na popsaných datových sadách. Z důvodu delšího trénování už nebyl rozsáhle testován vliv předzpracování na přesnost a namísto toho se prováděl test se základním předzpracováním a bez něj. Předzpracování zahrnuje odstranění stopslov, číslic a mnohočetných mezer, krátkých slov a interpunkce. Dále pak byly zkoušeny 2 formáty vstupu pro neuronové sítě, a to binární kódování a kódování pomocí matice TF-IDF. Obě tyto metody byly obsaženy ve třídě `Tokenizer`, která je součástí knihovny Keras. Testováno bylo 6 architektur, a to hustá síť, jednoduchá RNN, LSTM, obousměrná LSTM, GRU, konvoluční síť a spojení GRU a konvoluční sítě. Vzhledem k těžšímu hledání vhodné architektury a parametrů bylo některým modelům věnováno méně pozornosti a je možné, že uváděný výsledek není maximem, kterého je daná síť schopna dosáhnout.

7.1 Výsledky testů

V této sekci jsou popsány výsledky testů neuronových sítí nad jednotlivými sadami.

7.1.1 Reuters

V případě datové sady Reuters byla nadprůměrná přesnost nejčastěji při použití binárního kódování a s použitím předzpracování textu. Nejlepšího výsledku dosáhla konvoluční síť ve spojení s GRU sítí a to 96,41 %, avšak i ostatní sítě se pohybovaly kolem stejné přesnosti, kdy nejhorší byly RNN a obousměrná LSTM se stejnou přesností 95,79 % (Přesnost se shodovala na 7 desetinných čísel).

Tabulka 10: Přesnost jednotlivých neuronových sítí pro datovou sadu Reuters

Síť	Přesnost
Hustá	96,36 %
RNN	95,78 %
Obousměrná LSTM síť	95,79 %
Konvoluční síť s GRU	**96,41 %**
GRU	96,27 %
Konvoluční	95,96 %
LSTM	96,05 %

7.1.2 AG's News

I v případě AG's News byly nejlepší výsledky dosaženy s použitím binárního kódování, avšak tentokrát bez použití předzpracování textu. Nejlépe si vedla konvoluční síť, která svou přesností

88,74 % předčila druhou nejlepší síť o více jak 1 %. Nejhorší dvě sítě měly překvapivě opět stejnou přesnost a to 85,76 %, avšak nyní to byla hluboká a obousměrná síť.

Tabulka 11: Přesnost jednotlivých neuronových sítí pro datovou sadu AG's News

Síť	Přesnost
Hustá	85,76 %
RNN	86,75 %
Obousměrná LSTM síť	85,76 %
Konvoluční síť s GRU	86,75 %
GRU	86,09 %
Konvoluční	**88,74 %**
LSTM	85,43 %

7.1.3 DBpedia

U této datové sady dosahovaly sítě velmi dobré přesnosti 97 %, a to při použití binárního kódování a bez předchozího předzpracování textu. Nejpřesnější byla opět konvoluční síť s přesností 98,06 %. Ale i nejméně přesná síť dosahovala velmi dobré přesnosti 97,05 %. Tato vysoká přesnost mohla být způsobena faktem, že texty v této datové sadě neobsahují tolik chyb a nespisovných výrazů, a navíc je to jedna z největších testovaných datových sad.

Tabulka 12: Přesnost jednotlivých neuronových sítí pro datovou sadu DBpedia

Síť	Přesnost
Hustá	97,24 %
RNN	97,26 %
Obousměrná LSTM síť	97,29 %
Konvoluční síť s GRU	97,06 %
GRU	97,38 %
Konvoluční	**98,06 %**
LSTM	97,24 %

7.1.4 20 Newsgroups

Tato datová sada se ukázala jako nejhůře klasifikovatelná, co se anglických sad týče. Přesnosti zřídka přesáhly 70 % a bylo zapotřebí použít mnoho dalších technik jako přidání šumu, použití dropout vrstvy a přidání normalizačních vrstev. Po aplikaci zmíněných technik si nejlépe vedla obousměrná síť se 72,18 % a to při použití TF-IDF a bez dalšího předzpracování textu. Naopak nejhůře si vedla konvoluční síť s pouhými 64,63 %. Bylo překvapivé, že u této sady síť LSTM reagovala pozitivně na velmi velké procento šumu i při 60 % byla přesnost větší než při nižších hodnotách šumu.

Tabulka 13: Přesnost jednotlivých neuronových sítí pro datovou sadu 20 Newsgroups

Síť	Přesnost
Hustá	69,64 %
RNN	68,95 %
Obousměrná LSTM síť	**72,18 %**
Konvoluční síť s GRU	69,24 %
GRU	70,69 %
Konvoluční	64,63 %
LSTM	71,99 %

7.1.5 Yelp

V případě Yelp byly výsledky, nehledě na způsob předzpracování i použité kódování, téměř totožné a rozdíly se pohybovaly v rámci desetin procent. S velmi malým rozdílem bylo nejpřesnější použití metody TF-IDF spolu s předzpracováním textu. Překvapivě největší přesnost měla základní rekurentní síť s 92,75 %. Nejmenší přesnost měla konvoluční síť kombinovaná s LSTM sítí, která dosáhla přesnosti 91,92 %.

Tabulka 14: Přesnost jednotlivých neuronových sítí pro datovou sadu Yelp

Síť	Přesnost
Hustá	92,17 %
RNN	**92,75 %**
Obousměrná LSTM síť	92,49 %
Konvoluční síť s GRU	91,93 %
GRU	92,64 %
Konvoluční	92,02 %
LSTM	92,65 %

7.1.6 ČSFD

U této datové sady sítě dosahovaly přesnosti kolem 50 %. Byly použity sady, kde témata jednotlivých popisků mohla obsahovat více slov, což mohlo způsobit, že síť dobře odhalila správně část tématu, ale kvůli přísnému označení bylo toto označení bráno jako chyba. Proto byla vytvořena reprezentace tématu pomocí vektoru, kde víceslovné názvy byly převedeny na vektor, kde témata, která se v popisu nevyskytovala, obsahovala číslo 0 a číslo 1 pokud se téma v popisku vyskytovalo. Tento způsob snížil počet témat z 10 na 7, avšak přesnost se zvýšila pouze o 15 %, takže je otázkou, jestli zvýšená přesnost byla způsobená vhodnější reprezentací témat, anebo snížením počtu témat. Experiment provedený s datovou sadou 20 newsgroups v kapitole 6.3.4, kde bylo 20 témat sníženo na 7 naznačuje, že přesnost se zvýšila spíše díky jinému formátu témat než díky snížení počtu kategorií.

Nejlepších výsledků bylo dosaženo s binárním kódováním a bez použití předzpracování. Nejlepší přesnost měla obousměrná síť s 52,33 %, zatímco nejhůře si vedla konvoluční síť s 46,22 %.

V případě vektorové reprezentace tématu nejlépe fungovalo použití TF-IDF tentokrát s použitím předzpracování. Nejlepší přesnost měla síť LSTM 66,62 % a nejhůře si vedla opět konvoluční síť 60,02 %. Rozdíly v přesnosti mezi jednotlivými sítěmi jsou podobné jako u reprezentace témat bez použití vektorů.

Tabulka 15: Přesnost jednotlivých neuronových sítí pro datovou sadu ČSFD

Síť	Přesnost (Jedna kategorie)	Přesnost (Vektorová kategorie)
Hustá	49,76 %	62,12 %
RNN	50,15 %	65,44 %
Obousměrná LSTM síť	**52,33 %**	66,17 %
Kovnoluční síť s GRU	50,77 %	66,00 %
GRU	51,39 %	64,20 %
Konvoluční	46,22 %	60,02 %
LSTM	51,47 %	**66,62 %**

7.2 Porovnání s klasickými metodami

Porovnání výsledků klasických a hlubokých metod ukazuje, že neuronové sítě jsou ve většině případů lepšími klasifikátory, kromě datových sad 20 Newsgroups a Yelp, avšak zde se rozdíl v přesnostech liší pouze o 0,25 % v obou případech. Tudíž v případě potřeby dosažení nejpřesnějších výsledků jsou neuronové sítě lepší volbou. Avšak pokud je faktorem i čas učení a paměťová náročnost, jsou klasické metody schopny velmi dobře konkurovat, obzvláště model Naivní Bayes, který vždy dosáhl velmi vysoké přesnosti, navíc v nejkratším čase.

V tabulce 16 jsou uvedeny nejelpší výsledky dosažené pomocí klasických a hlubokých modelů. Za každým výsledkem je v závorce uveden model, který dané přesnosti dosáhl. V rámci neuronových sítí jsou pro modely použity zkratky. BL značí obousměrnou rekuretní síť využívající LSTM, Konv je konvoluční síť a KG je zkratka pro konvoluční síť spojenou se sítí GRU.

Tabulka 16: Porovnání přesností klasických modelů s modely neuronových sítí vzhledem k datové sadě

Dataset	Klasický model	Hluboký model
Reuters	95,30 % (NB)	**96,41%** (KG)
AGs-News	86,75 % (SVM)	**88,74 %** (Konv)
DBpedia	97,59 % (SVM)	**98,06 %** (Konv)
20 Newsgroups	**72,43 %** (NB)	72,18 % (BL)
Yelp	**93,00 %** (SVM)	92,75 %(RNN)
ČSFD	51,63 % (SVM)	**52,33 %** (BL)

7.3 Využití vnořené vrstvy

Byly prováděny i testy s vnořenou vrstvou (Embedding layer). Bohužel v případě testování této vrstvy často docházelo k různým chybám v knihovně TensorFlow a nepodařilo se najít přesnou příčinu, která by dané chyby způsobovala. I přes tyto chyby se podařilo několik testů dokončit, ale protože nebyly vyzkoušeny všechny možné způsoby předzpracování jsou tyto výsledky zmíněny v této samostatné kapitole.

Byly vyzkoušeny 3 architektury vnořených vrstev všechny ve spojení s LSTM vrstvou. První dvě architektury využívaly vnoření GloVe s tím, že v jedné byla vrstva vnoření trénována a v té druhé nikoliv. Nakonec třetí architektura byla trénována bez předchozího nastavení vah. Ukázalo se, že použití vrstvy vnoření podává u menších datových sad spíše průměrné výsledky. Avšak u sad Yelp a DBpedia dosáhly tyto architektury nejlepších výsledků, a to 98,37 % u DBpedia (při použití formátu TF-IDF bez předzpracování), kde bylo maximum bez vnoření 98,06 %. V případě Yelp bylo dosaženo přesnosti 92,97 % (opět TF-IDF tentokrát s použitím předzpracování), což je nepatrně lepší přesnost oproti 92,75 %. V obou případech šlo o architekturu, která využívala GloVe a bylo jí dovoleno vrstvu vnoření dále trénovat.

8 Extrakce topiků pomocí autoenkodéru

Pro extrakci topiků, neboli slov významných pro dané téma, se ve valné většině případů používá LDA 2.3. V době psaní této práce nebyly nalezeny žádné experimenty o využití neuronové sítě pro nalezení topiků. Právě nalezení topiků pomocí neuronové sítě je cílem této kapitoly, kde bude popsán základní model pro hledání topiků, způsob získání topiků a následné porovnání získaných topiků s topiky získanými pomoci LDA metody.

8.1 Základní model

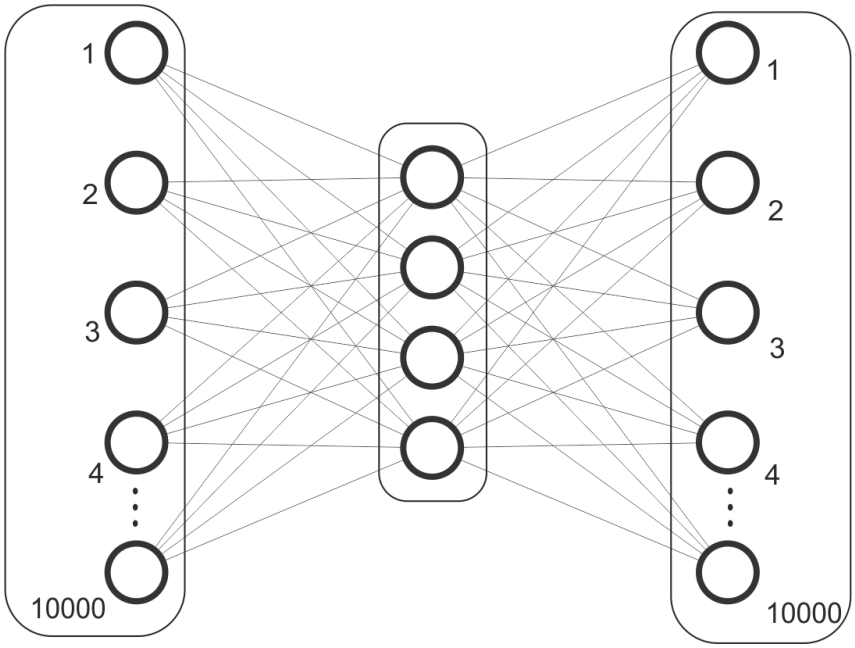
Základní model byl jednoduchý autoenkodér s hustě propojenými vrstvami, který měl jenom jednu skrytou vrstvu, kde počet neuronů v této vrstvě odpovídal počtu unikátních témat v dané datové sadě. Vstupní a výstupní vrstvy měly stejný počet neuronů, který odpovídal počtu slov, se kterými měla neuronová síť pracovat. Na obrázku 15, je uveden příklad modelu, kde sada měla 4 témata a počet slov, se kterýma se pracovalo byl 10000. Bylo pracováno s různými formáty vstupu, a to s binárním formátem, kdy výstupní vrstva používala aktivační funkci sigmoid a ztrátová funkce byla binární křížová entropie, a také se vstupy zpracovanými pomocí metody TF-IDF. Jako optimalizér byl použit Adam.

8.2 Získání topiků

Po naučení modelu byly získány váhy enkodéru a dekodéru, kde váhy matice enkodéru byly transponovány, aby se dimenze matice enkodéru shodovala s maticí dekodéru. Tyto dvě matice sloužily jako samostatné topikové modely a byly analyzovány zvlášť. Ke každému řádku matice byla přiřazena slova z naučeného tokenizéru, aby bylo možné vědět, k jakému slovu patří jaká hodnota z matice vah. Následně byly hodnoty matice vah spolu s náležitými slovy setříděny od největšího po nejmenší. Z těchto setřízených řádků bylo následně extrahováno 20 slov s největší hodnotou váhy a 20 slov s nejmenší hodnotou váhy (u nejmenších vah se neočekávalo, že budou obsahovat relevantní slova, ale byla extrahována, aby se tento předpoklad mohl potvrdit). 20 extrahovaných slov každého řádku představovalo 20 slov, která by měla nejvíce identifikovat dané téma. Spolu s 2 modely, využívajícími matice vah autoenkodéru, byl vytrénován také model LDA, který se často používá pro extrakci topiků. Z modelu LDA bylo obdobně extrahováno 20 nejvýraznějších slov pro jednotlivá témata a tato slova byla stanovena jako základní hranice, kterou by měl experimentální model překonat nebo se k ní alespoň přiblížit. Samotná kvalita extrahovaných slov jednotlivých témat se měřila pomocí koherence, kvality shlukování a porovnáním slov.

Koherence, zmíněná v kapitole 4.2, se často používá pro zhodnocení kvality topiků jednotlivých témat. V rámci testování byly použity obě zmíněné metriky tedy UCI i UMASS.

Model topiku LDA je schopen rozdělit dokumenty do jednotlivých shluků, na základě naučených témat. V případě, že je známo skutečné téma, kterému daný dokument náleží, je možné



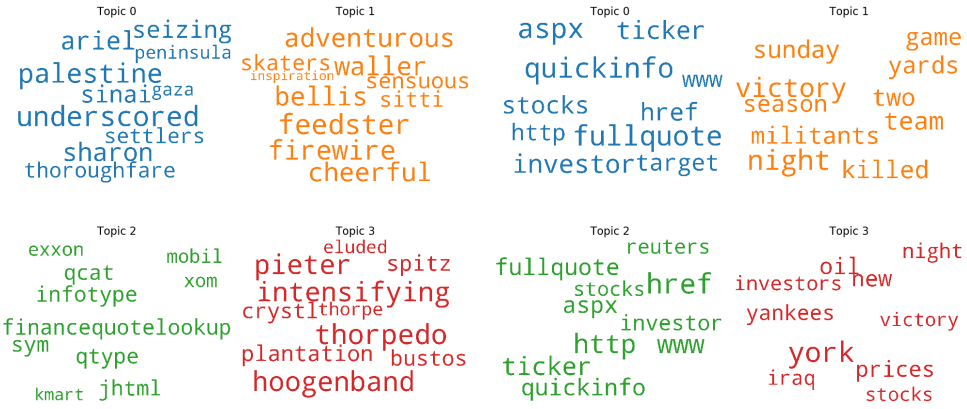
Obrázek 15: Architektura autoenkodéru pro 4 témata

změřit, jak přesný je model ve shlukování. Toto ověření přesnosti bylo použito i na neuronový model topiků, kde bylo cílem ověřit, jestli neuronový model umí lépe provádět shlukování. Způsob shlukování pomocí neuronové sítě je popsán v kapitole 8.6.1

U kombinace koherence a měření kvality shlukování se ukázalo, že tato dvě měřítka se dobře doplňují. Když vyjde dobré skóre u koherence, avšak kvalita shlukování je nízká, většinou to znamená, že model našel dobře slova menšího počtu topiků. Například u kvalitní koherence s kvalitou shlukování 30 % při 4 tématech znamenalo, že model dobře našel topiky pro jedno téma ze 4. Bylo možné, že vznikl i opačný případ, kdy byla kvalita shlukování vysoká, ale skóre koherence nízké, což mohlo znamenat, že model více rozprostřel náležitost slov jednotlivým tématům, ale nezachytil ta hlavní slova. Ukázka těchto případů je na obrázcích 16a a 16b.

Nakonec porovnání slov bylo použito u těch modelů, které měly dobré výsledky v předchozích dvou faktorech. Zde je potřeba na posouzení člověka, zda-li jsou nalezená slova kvalitní a právě proto byl použit filtr předchozích dvou faktorů.

Hlavními faktory, které měly vliv na kvalitu nalezených topiků, byla zvolená metoda regularizace, její intenzita a počet epoch. Bylo také experimentováno s reprezentací dokumentů jak v binární formě, tak pomocí metody TF-IDF. Okrajově bylo experimentováno s šumem a použitím techniky dropout, ale u těchto vrstev se ukázalo, že výsledky spíše zhoršují. Nakonec ještě bylo experimentováno s velikostí dávky a s normalizací vah, a to po každé epoše anebo po každé dávce.



(a) Dobrá kvalita shlukování nízká koherence

(b) Nízká kvalita shlukování dobrá koherence

Obrázek 16: Ukázka kvality slov při vysoké kvalitě shlukování a nízkém skóre koherence a opačně vysokém skóre koherence a nízké kvalitě shlukování.

8.2.1 Vliv počtu epoch na kvalitu modelu

Větší počet epoch zpravidla zlepšoval kvalitu modelu. Při použití časného zastavení, což je zastavení trénovaní, pokud nedojde k požadované změně do určitého počtu epoch, bylo dosaženo největší kvality shlukování. Avšak pokud byly načteny váhy, které dosáhly nejnižší hodnoty ztrátové funkce, nevedlo to k tak dobrým výsledkům, jako když byly ponechány váhy po poslední natrénované epoše. Počet epoch potřebných pro naučení se lišil podle velikosti datové sady a zpravidla čím větší sada, tím méně epoch bylo potřeba.

Z tabulky 17 je vidět, že výsledky jsou vždy lepší pro výstupní váhy, což je pravděpodobně způsobeno metodou regularizace, jejíž vliv je popsán v následující kapitole. Také paradoxně dosáhla síť nejlepšího skóre koherence při trénování po 100 epochách. Ale kvalita shlukování je zde naopak nejlepší, a i při pohledu na slova s nejvyšší váhou jednotlivých témat je vidět, že síť našla topiky jednoho až dvou témat a ne požadovaných čtyř. Součástí výsledků je i aplikovaní brzkého zastavení, což už je metoda regularizace, ale vzhledem k tomu, že se zde pracuje s počtem epoch byly tyto výsledky zařazeny do této sekce.

8.2.2 Vliv regularizace na kvalitu modelu

Regularizace měla velký vliv na kvalitu nalezených topiků, a proto se na ni testování nejvíce zaměřovalo. Byly vyzkoušeny metody L1, L2 a spojení metod L1 a L2. Dále byla regularizace aplikována na vstupní váhy, výstupní váhy a na obě váhy současně. Nakonec byl zkoumán vliv různých typů regularizací, nabízených knihovnou Keras, a to regularizace aktivační funkce, kernelu a biasu.

Tabulka 17: Vliv počtu epoch na kvalitu modelu topiků

Počet epoch	UCI	UMASS	Kvalita shlukování
100 (in)	-16,31	-20,12	39,66 %
100 (out)	**-0,47**	**-2,12**	41,11 %
500 (in)	-13,52	-20,45	53,37 %
500 (out)	-1,71	-3,34	62,62 %
1000 (in)	-11,61	-17,76	37,51 %
1000 (out)	-3,76	-6,19	60,96 %
Zastavení v bodě (in)	-13,94	-19,52	47,36 %
Zastavení v bodě (out)	-2,77	-4,84	67,08 %
Zastavení 100 epoch po bodu (in)	-13,77	-20,15	45,86 %
Zastavení 100 epoch po bodu (out)	-4,42	-6,05	**68,47 %**

V tabulkách níže jsou uvedeny vybrané výsledky po aplikaci regularizace na jednotlivé vrstvy. Slova v závorce (in, out, both) určují, o kterou vrstvu se jedná. Slovo v závorce před názvem metody regularizace udává, zda-li byla metoda aplikována na vstupní nebo výstupní váhy nebo na obě dvě. Druhé slovo v závorce na konci názvu (in, out) udává, zda-li se jedná o výsledek modelu topiků vytvořeného z vah vstupní nebo výstupní vrstvy.

Tabulka 18 ukazuje vliv použití regularizace na aktivační funkci. Jsou zde uvedeny pouze lepší výsledky. Celá tabulka je dostupná v příloze této práce. Nejvíce se osvědčilo použití regularizace L1 spolu s L2. Samotná metoda L1 vzácně pomohla vytvořit dobrý model. L2 vytvářela spíše průměrné modely, ale měly relativně dobrou kvalitu shlukování. Převážně vytvořila kvalitní model matice vah výstupní vrstvy. Regularizace pouze výstupní vrstvy ke zlepšení modelu nevedla, naopak regularizace pouze vstupní vrstvy pomáhala dosahovat dobrých výsledků. Avšak nejlepšího výsledku bylo dosaženo použitím regularizace L1 spolu s L2 jak na vstupní, tak na výstupní vrstvě.

Tabulka 19 udává vliv regularizace kernelu. Tento druh regularizace vracel kvalitní modely, co se topiků týče, ale podle kvality shlukování tyto topiky zachytávaly pouze jedno téma. Použití regularizace L2 na výstupní vrstvu způsobilo vytvoření nejlepšího modelu topiků (paradoxně ze vstupních vah) v rámci regularizace kernelu.

Také bylo provedeno několik testů, kde byla použita regularizace na aktivační funkci a současně i kernel. Použity byly ty druhy regularizace, které samostatně dosahovaly nejlepších výsledků. Z osmi různých testů žádný nedosáhl dobrých výsledků.

Nakonec byl ještě vyzkoušen vliv regularizace biasu, kde přesnost shlukování modelu nikdy neklesla pod 50 %, ale samotné topiky pro jednotlivá témata nebyly příliš kvalitní.

8.2.3 Vliv různé reprezentace dokumentů

Použití TF-IDF namísto binárního formátu nepřineslo zlepšení výsledku a modely měly spíše průměrnou kvalitu. Nejlepšího výsledku bylo dosaženo použitím regularizace L1 a L2 na obě

Tabulka 18: Vliv regularizace aktivační funkce na kvalitu topiků

Způsob regularizace	UCI	UMASS	Kvalita shlukování
Bez regularizace (in)	-13,00	-16,50	56,82 %
Bez regularizace (out)	-11,80	-16,26	52,38 %
(in) L12 0,0001(out)	-5,32	-7,33	69,96 %
(in) L12 0,001(out)	**-2,86**	-4,68	76,00 %
(in) L1 0,001(out)	-5,73	-6,22	74,30 %
(both) L12 0,0001(out)	-2,93	**-4,52**	**76,48 %**
(both) L12 0,001(out)	-5,56	-7,40	47,56 %
(both) L12 0,01(out)	-3,43	-4,71	59,08 %
(both) L1 0,001(out)	-8,70	-9,30	70,61 %
(out) L1 0,001(out)	-14,34	-17,37	60,64 %
(both) L2 0,001(out)	-8,64	-11,12	50,99 %
(in) L2 0,001(out)	-8,51	-10,67	60,96 %
(out) L2 0,001(in)	-14,33	-18,69	63,95 %

Tabulka 19: Vliv regularizace kernelu na kvalitu topiků

Způsob regularizace	UCI	UMASS	Kvalita shlukování
(in) L12 0,0001 (out)	-1,36	-2,09	30,32 %
(in) L12 0,001 (out)	**-1,23**	-2,17	31,49 %
(out) L12 0,0001 (in)	-3,04	**-1,91**	45,24 %
(out) L12 0,0001 (out)	-4,45	-2,33	40,69 %
(in) L2 0,001 (in)	-4,80	-5,51	55,27 %
(in) L2 0,001 (out)	-4,67	-7,16	61,49 %
(out) L2 0,001 (in)	-2,41	-4,71	74,34 %
(out) L2 0,001 (out)	-6,40	-8,76	**74,89 %**

vrstvy při použití výstupních vah. Nejlepšího skóre koherence dosáhla síť s použitím regularizace L1 a L2 na vstupní vrstvu s použitím výstupních vah, avšak zde byla kvalita shlukování jenom 43,92 %.

Tabulka 20: Vliv reprezentace dokumentů pomocí TFIDF

Způsob regularizace	UCI	UMASS	Kvalita shlukování
(both) L2 0.001 (in)	-4.39	-6.14	53.11 %
(both) L2 0.001 (out)	-6.77	-6.48	56.76 %
(both) L12 0.0001 (out)	-6.00	-8.29	58.86 %
(both) L12 0.001 (out)	-6.32	-9.00	60.34 %
(both) L12 0.01 (out)	-6.15	-8.13	**70.19 %**
(in) L12 0.001 (out)	**-1.36**	**-2.08**	43.92 %
(in) L12 0.001 (in)	-3.67	-4.39	38.11 %
(in) L12 0.001 (out)	-5.49	-7.79	43.40 %

8.3 Porovnání s metodou LDA

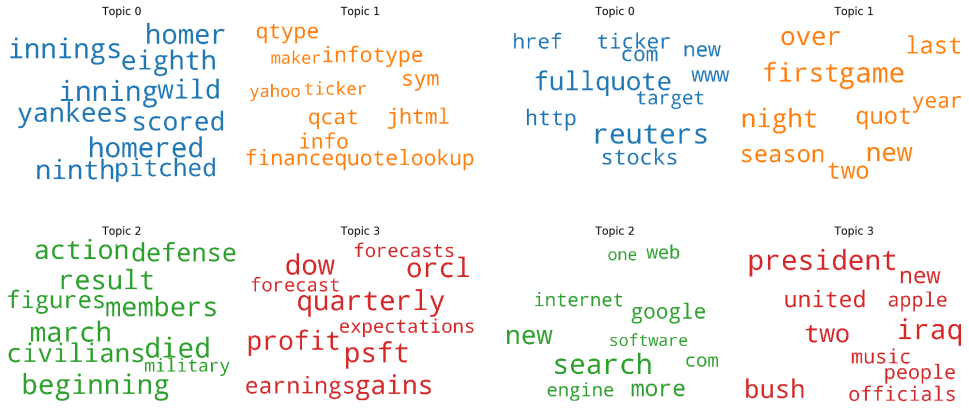
Po analýze různých vlivů na kvalitu topiků neuronového modelu byl nakonec nejlepší model porovnán s modelem LDA. Ukázalo se, že LDA je stále schopen vytvořit lepší model topiků, než model z neuronové sítě, a také je tohoto výsledku schopen dosáhnout značně rychleji. Na druhou stranu se ukázalo, že neuronový model je schopen dosáhnout lepší kvality shlukování. Na obrázku 17 jsou ukázány nejlepší nalezené topiky u sady AGs News. Témata této datové sady jsou svět, sport, obchod, technologie.

Tabulka 21: Porovnání LDA s neuronovým modelem topiků

Způsob regularizace	UCI	UMASS	Kvalita shlukování
LDA	-0,40	-2,59	69,94 %
Neurální model	-2,93	-4,52	76,48 %

8.4 Výsledky u ostatních datových sad

Jak už bylo zmíněno, tak všechny experimenty byly prováděny na datové sadě AGs News, díky tomu, že měla pouze 4 témata, což usnadňovalo kontrolu topiků. V rámci této sekce jsou uvedeny výsledky modelu u ostatních sad spolu s výsledky získané s metodou LDA. Bohužel u žádné datové sady nebyl neuronový model lepší než LDA model, avšak jeho výsledky nebyly špatné vezmeme-li v potaz, že architektura neuronové sítě (včetně metod regularizace) reaguje na každou datovou sadu jinak a při těchto testech byla ponechána stejná architektura u všech datových sad. I přes toto omezení dokázala podat přijatelné výsledky.



(a) Neuronový model (b) LDA

Obrázek 17: Ukázka topiků neuronového modelu a modelu LDA pro sadu AGs News

Tabulka 22: Kvalita modelu u ostatních datových sad

Způsob regularizace	UCI	UMASS	Kvalita shlukování
Reuters	-9,35	-14,51	58,07 %
Reuters (LDA)	0.19	-1.98	51,11 %
Yelp	0,51	-3,15	31,10 %
Yelp (LDA)	0.28	-2.10	47,07 %
DBpedia	-1,07	-4,83	50,53 %
DBpedia (LDA)	0.56	-2.56	68,11 %
20 newsgroups	-10.73	-17.80	19,18 %
20newsgroups (LDA)	-3.37	-8.05	25,56 %
AGs News	-2,93	-4,52	76,48 %
AGs News (LDA)	-0,40	-2,59	69,94 %

8.5 Výsledky na české datové sadě

Vzhledem k tomu, že optimalizace parametrů sítě byla prováděna na anglické sadě, dalo by se čekat, že při použití na českou sadu nebude model dávat dobré výsledky. Opak je ale pravdou. I zde sice nepřekonal model LDA, ale ani se jeho výsledkům příliš nevzdálil. Navíc se ukázalo, že model dokázal rozeznat slovenštinu od češtiny, kde skupina topiků, kde byla slovenská slova, obsahovala minimum českých slov a obráceně. Pro lepši zobrazení extrahovaných topiků byla sada ČSFD redukována na 4 témata. Na obrázku 18 jsou ukázány nejlepší nalezené topiky u sady ČSFD se 4 tématy. Jako témata této zmenšené sady byla vybrána romantika, komedie, akce a dokument.

(a) Neuronový model (b) LDA

Obrázek 18: Ukázka topiků neuronového modelu a modelu LDA pro sadu ČSFD

8.6 Shlukování dokumentů

Jelikož se model LDA dá použít pro shlukování dokumentů do jím náležících témat, byl ke stejnému účelu naprogramován i neuronový model topiků. V této podkapitole je popsán způsob získání tématu, ke kterému analyzovaný dokument náleží a také způsob míry přesnosti neuronového modelu topiků.

8.6.1 Přiřazení indexu shluku dokumentu

Při zjišťování náležitosti daného dokumentu do jednotlivého shluku se nejdříve slova z dokumentu převedou do vektoru o N prvcích, kde N je počet slov, které se tokenizátor naučil v době tréninku. Tento vektor bude obsahovat jedničky u slov, která se dostatečně často vyskytovala v trénovací sadě (nejčastějších N slov). Vytvořený vektor se poté vynásobí s jednotlivými řádky váhové matice a hodnoty po tomto vynásobení se poté sečtou do jednoho čísla. Vynásobením a následným sečtením jednoho řádku z matice vah neuronového modelu se zjišťuje, jakou měrou

daný dokument patří do daného, shluku, resp. tématu. Nakonec je vybrán index toho shluku, u kterého vyšla nejvyšší hodnota.

8.6.2 Způsob měření přesnosti shlukování

Měření kvality shlukování bylo prováděno stejně jako v případě testování modelu LDA popsaného v sekci 6.1.1. Tedy vybraly se všechny pravděpodobnosti náležitostí u jednotlivých témat, tyto pravděpodobnosti se setřídily od největšího po nejmenší a poté se postupně vybíraly doposud nepoužité indexy, které byly následně přiřazeny k jednotlivým topikům.

9 Závěr

Testy ukázaly, že klasické modely stále mohou konkurovat neuronovým sítím v případě základní klasifikace textu. Je snazší nastavit jejich parametry a také doba trénování je mnohem kratší v porovnání s neuronovými sítěmi. Avšak pokud je třeba dosáhnout nejlepších výsledků, jsou neuronové sítě lepší volbou. Dále se ukázalo, že předzpracování ne vždy může přispět ke zlepšení přesnosti klasifikace a hodně záleží, jak kvalitní je jazyk v datové sadě. Nakonec bylo ukázáno, že neuronová síť je schopna vytvořit model topiků, avšak tento model stále není tak kvalitní, jako model získaný pomocí metody LDA, avšak v rámci shlukování byl neuronový model v určitých případech schopen dosáhnout lepších výsledků.

Nebyla zcela vyřešena otázka, jak správně testovat model LDA bez toho, aby naučené indexy musel ručně přiřazovat člověk. Také zůstává otázkou, jestli by složitější architektura autoenkodéru mohla pomoci k získání kvalitnějších topiků.

Odkazy

[1] *7 Types of Neural Network Activation Functions: How to Choose?* URL: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/ (cit. 13. 05. 2020).

[2] Charu Aggarwal. *Machine learning for text*. Cham, Switzerland: Springer, 2018. ISBN: 978-3-319-73530-6.

[3] David M. Blei. „Probabilistic topic models“. In: *Communications of the ACM*. Sv. 55. 4. 2012, s. 77–84. DOI: 10.1145/2133806.2133826. URL: https://dl.acm.org/doi/10.1145/2133806.2133826.

[4] Leo Breiman a Adele Cutler. *Random Forests Leo Breiman and Adele Cutler*. URL: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm (cit. 13. 05. 2020).

[5] Jason Brownlee. *What Are Word Embeddings?* URL: https://machinelearningmastery.com/what-are-word-embeddings/ (cit. 24. 01. 2020).

[6] *Convolutional Neural Networks (CNNs / ConvNets)*. URL: https://cs231n.github.io/convolutional-networks/ (cit. 14. 05. 2020).

[7] František Čermák. *Jazyk a jazykověda. přehled a slovníky*. Vyd. 3., dopl. Praha: Karolinum, 2001. ISBN: 80-246-0154-0.

[8] *Česko-Slovenská filmová databáze*. URL: https://www.csfd.cz/ (cit. 13. 05. 2020).

[9] Francois Chollet. *A ten-minute introduction to sequence-to-sequence learning in Keras*. URL: https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html (cit. 24. 01. 2020).

[10] François Chollet. *Deep learning with Python*. Shelter Island, NY: Manning, 2018. ISBN: 978-161-7294-433.

[11] Junyoung Chung et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. eprint: arXiv:1412.3555. URL: https://arxiv.org/abs/1412.3555.

[12] Jakub Jirutka. *Csfd-parser 1.0.5*. 2013. URL: https://pypi.org/project/csfd-parser/ (cit. 13. 05. 2020).

[13] *Keras*. URL: https://www.tensorflow.org/guide/keras (cit. 06. 05. 2020).

[14] Mahendru Khyati. *A Detailed Guide to 7 Loss Functions for Machine Learning Algorithms with Python Code*. 2019. URL: https://www.analyticsvidhya.com/blog/2019/08/detailed-guide-7-loss-functions-machine-learning-python-code/ (cit. 13. 05. 2020).

[15] Simeon Kostadinov. *How Recurrent Neural Networks work*. 2017. URL: https://towardsdatascience.com/learn-how-recurrent-neural-networks-work-84e975feaaf7 (cit. 14. 05. 2020).

[16] David D. Lewis. *Reuters-21578, Distribution 1.0*. 1997. URL: http://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection (cit. 07. 05. 2020).

[17] Marc Moreno Lopez a Jugal Kalita. *Deep Learning applied to NLP*. 2017. eprint: arXiv:1703.03091. URL: https://arxiv.org/abs/1703.03091.

[18] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781. URL: https://arxiv.org/abs/1301.3781.

[19] Tom Mitchell. *Twenty Newsgroups Data Set*. 1999. URL: https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups (cit. 07. 05. 2020).

[20] Gupta Prashant. *Decision Trees in Machine Learning*. 2017. URL: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052 (cit. 13. 05. 2020).

[21] Mohammed Rampurawala. *Classification with TensorFlow and Dense Neural Networks*. 2019. URL: https://heartbeat.fritz.ai/classification-with-tensorflow-and-dense-neural-networks-8299327a818a (cit. 14. 05. 2020).

[22] Sunil Ray. *Understanding Support Vector Machine(SVM) algorithm from examples (along with code)*. URL: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/ (cit. 07. 05. 2020).

[23] Michael Röder, Andreas Both a Alexander Hinneburg. „Exploring the Space of Topic Coherence Measures“. In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. WSDM '15. Shanghai, China: Association for Computing Machinery, 2015, s. 399–408. ISBN: 9781450333177. DOI: 10.1145/2684822.2685324. URL: https://doi.org/10.1145/2684822.2685324.

[24] Radim Řehůřek. *Gensim - topic modelling for humans*. 2009. URL: https://radimrehurek.com/gensim/index.html (cit. 14. 05. 2020).

[25] Radim Řehůřek. *Similarity Queries*. URL: https://radimrehurek.com/gensim/tut3.html (cit. 06. 05. 2020).

[26] Doshi Sanket. *Various Optimization Algorithms For Training Neural Network*. 2019. URL: https://medium.com/@sdoshi579/optimizers-for-training-neural-network-59450d71caf6 (cit. 13. 05. 2020).

[27] *Scikit-learn Machine Learning in Python*. URL: https://scikit-learn.org/stable/ (cit. 14. 05. 2020).

[28] Jain Shubham. *An Overview of Regularization Techniques in Deep Learning (with Python code)*. 2018. URL: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/ (cit. 07. 05. 2020).

[29] *TensorFlow*. URL: https://www.tensorflow.org/ (cit. 14. 05. 2020).

[30] *Tf–idf*. URL: https://en.wikipedia.org/wiki/tf-idf (cit. 14. 05. 2020).

[31] Sachin Tyagi. *LDA: NLP and Code Analysis – Part II*. 2015. URL: https://blog.imaginea.com/lda-nlp-and-code-analysis/ (cit. 06. 05. 2020).

[32] Michael Walker. *Random Forests Algorithm*. 2013. URL: https://www.datasciencecentral.com/profiles/blogs/random-forests-algorithm (cit. 13. 05. 2020).

[33] Serdar Yegulalp. *What is TensorFlow? The machine learning library explained*. 2019. URL: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html (cit. 07. 05. 2020).

[34] *Yelp Dataset*. 2018. URL: https://www.kaggle.com/yelp-dataset/yelp-dataset/metadata (cit. 07. 05. 2020).

[35] Xiang Zhang. *Datasety AG News a DBpedia*. 2015. URL: http://xzh.me/ (cit. 07. 05. 2020).

10 Přílohy

- **TextAnalysis** - složka se zdrojovými kódy a datovými sadami
  - **classic_methods** - složka obsahující třídy, které pracují s klasickými metodami (NB, SVM, LDA atd.)
    - **methods** - třídy pracující s klasickými metodami
    - **tests** - skripty pro testování jednotlivých modelů
    - **preprocessor** - skripty pro předzpracování textu, použité pro testování vlivu předzpracování na přesnost a načítání datových sad. Tyto skripty byly nahrazeny třídou Dataset_Helper, ale byly ponechány z důvodů kompatibility
    - **run** - skript provádějící testování klasifikace jednotlivých datových sad a modelů
    - **RunForModelSettingsTest** - skript pro nalezení nejlepších parametrů u modelů LDA a LSA
  - **dataset_loader** - složka obsahující skripty, které se starají o správné načtení a předzpracování datových sad
  - **datasets** - složka obsahující analyzované datové sady
  - **neural_networks** - složka obsahující skripty, se kterými byly prováděny experimenty
    - **models** - třídy představující jednotlivé architektury neuronových sítí
    - **readme.txt** - soubor popisující účel jednotlivých skriptů v této složce
  - **text_generators** - složka obsahující generátory textu, které byly použity ve spojení s neuronovými sítěmi.
  - **readme.md** - soubor popisující celý projekt. Také obsahuje instrukce, popisující, jak naistalovat potřebné knihovny pro provádění experimentů.
  - **results_saver.py** - skript obsahující metody pro ukládání výsledků experimentů
- **Architektury** - složka obsahující obrázky a json reprezentace architektur vygenerované pomocí knihovny Keras použité při experimentech s klasifikací
- **Nalezené topiky** - složka obsahující obrázky topiků nalezených ve všech analyzovaných datových sadách
- **Tvorba datovych sad** - složka obsahující skripty použité pro vytvoření datových sad Yelp a ČSFD
- **Tabulky.pdf** - PDF dokument obsahující tabulky výsledků modelu topiků pro neuronový model